Winter 2002

Video Indexing and Retrieval Techniques Using Novel Approaches to Video Segmentation, Characterization, and Similarity Matching

Waleed Ezzat Farag
Old Dominion University

VIDEO INDEXING AND RETRIEVAL TECHNIQUES USING NOVEL APPROACHES TO VIDEO SEGMENTATION, CHARACTERIZATION, AND SIMILARITY MATCHING

by

Waleed Ezzat Farag
M.Sc. Computer Engineering, April 1997, Zagazig University, Egypt
B.Sc. Electronics and Communications Engineering, May 1993, Zagazig University, Egypt

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2002

Approved by:

Hussein Abdel-Wahab (Director)

Kurt Maly (Member)

Shunichi Toida (Member)

Christian Wild (Member)

Lee Belfore (Member)

ABSTRACT

VIDEO INDEXING AND RETRIEVAL TECHNIQUES USING NOVEL APPROACHES TO VIDEO SEGMENTATION, CHARACTERIZATION, AND SIMILARITY MATCHING

Waleed Ezzat Farag
Old Dominion University, 2002
Director: Dr. Hussein Abdel-Wahab

Multimedia applications are rapidly spread at an ever-increasing rate introducing a number of challenging problems at the hands of the research community. The most significant and influential problem, among them, is the effective access to stored data. In spite of the popularity of keyword-based search technique in alphanumeric databases, it is inadequate for use with multimedia data due to their unstructured nature. On the other hand, a number of content-based access techniques have been developed in the context of image indexing and retrieval; meanwhile video retrieval systems start to gain wide attention. This work proposes a number of techniques constituting a fully content-based system for retrieving video data. These techniques are primarily targeting the efficiency, reliability, scalability, extensibility, and effectiveness requirements of such applications. First, *an abstract representation of the video stream*, known as the DC sequence, is extracted. Second, to deal with the problem of *video segmentation*, an efficient neural network model is introduced. The novel use of the neural network improves the reliability while the efficiency is achieved through the instantaneous use of the recall phase to identify shot boundaries. Third, the problem of *key frames extraction* is addressed using two efficient algorithms that adapt their selection decisions based on the amount of activity found in each video shot enabling the selection of a near optimal expressive set of key frames. Fourth, the developed system employs an *indexing scheme* that supports two low-level features, color and texture, to represent video data. Finally, we propose, in the retrieval stage, a novel model for performing *video data matching task* that integrates a number of human-based similarity factors. All our software implementations are in Java, which enables it to be used across heterogeneous platforms. The retrieval system performance has been evaluated yielding a very good retrieval rate and accuracy, which demonstrate the effectiveness of the developed system.

ACKNOWLEDGMENTS

Praise to Allah. the Lord of the universe. the most Merciful and Beneficent. without his blessings this research effort would not become a reality.

This thesis is the outcome of many efforts and collaborations among a number of people. Thus. I would briefly attempt to express my appreciation to all of them and hopefully I will not forget any one. First of all. I would like to express my sincere gratitude to my advisor. Prof. Hussein Abdel-Wahab. for his great help and support through out the whole period of this research effort. Starting from the first semester when I joined the Ph.D. program till this current moment. I always appreciate the time he assigned to me for a continuous and constructive guidance. His advices were always very useful and to the point. I also appreciate the time he took to promptly review and comment on all my technical reports. papers. and all the versions of this dissertation.

My appreciation also goes to all the members of my committee Dr. Maly. Dr. Toida. Dr. Wild. and Dr. Belfore for their help and guidance.

Thanks to all my professors (in particular Prof. Ibrahim Ziedan) back in my home country. Egypt. who taught me long time ago the ABCs of performing research.

I want to thank Dr. Nervana. my lovely wife. who was extremely patient and understanding during all these years. Nervana left her Medical job in Egypt to come here and accompany me during my study period to achieve one of the corner stone in our life. I am here just trying to express my great gratitude to her for all what she did and still doing and for her never fading love. Also I would like to thanks Jannah. my wonderful baby daughter. for her cheerful smiles that were the fuel that kept me always up and focused.

Finally. My great appreciation and gratitude to my parents (my father Ezzat Farag and my mother Mahaseen Shawer) without whom I would not be what I am right now. Their care and encouragement are always unstinted and I have always the feeling that whatever I did or will do for them will never be one tenth of what they offered me. Hopefully. these little words would express my sincere appreciation towards them.

TABLES OF CONTENTS

LIST OF TABLES

Table | Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Recently, multimedia applications are undergoing explosive growth due to the monotonic increase in the available processing power and bandwidth. This incurs the generation of large amounts of media data that need to be effectively and efficiently organized and stored. While these applications generate and use vast amounts of multimedia data, the technologies for organizing and searching them are still in their infancy. These data are usually stored in multimedia archives utilizing search engines to enable users to retrieve the required information.

Searching a repository of data is a well-known important task whose effectiveness determines, in general, the success or failure in obtaining the required information. A valuable experience that has been gained by the explosion of the web is that the usefulness of vast repositories of digital information is limited by the effectiveness of the access methods. In a nutshell, the above statement emphasizes the great importance of providing effective search techniques. For alphanumeric databases many portals [6] such as *google, yahoo, msn, and excite* have become widely accessible via the web. These search engines provide their users keyword-based search model in order to access the stored information but the inaccurate search results of these search engines is a known drawback.

For multimedia data, describing unstructured information (such as video) using textual terms is not an effective solution because they cannot be uniquely described by a number of statements. That is mainly due to the fact that human opinions vary from one person to another [3], so that two persons may describe a single image by totally different statements. Therefore, the highly unstructured nature of multimedia data renders keyword-based search techniques inadequate. Video streams are considered the most complex form of multimedia data because they contain almost all other forms such as images and audio in addition to their inherent temporal dimension. The central role of

---

The journal model for this dissertation is the *IEEE Transactions on Knowledge and Data Engineering*.

video data among all other multimedia forms motivated us to focus in this research on proposing an effective search paradigm for that particular media type.

One promising solution that enables searching multimedia data, in general, and video data in particular is the concept of content-based search and retrieval. The basic idea is to access video data by their contents; for example, using one of the visual content features. Realizing the importance of content-based searching, researchers have started investigating the issue and proposing creative solutions. Most of the proposed video indexing and retrieval prototypes have the following two major phases [31]:

- Database population phase consisting of the following steps:
    - Shot boundary detection. The purpose of this step is to partition a video stream into a set of meaningful and manageable segments [38], which then serve as the basic units for indexing.
    - Key frames selection. This step attempts to summarize the information in each shot by selecting representative frames that capture the salient characteristics of that shot.
    - Extracting low-level features from key frames. During this step, a number of low-level spatial features (color, texture, etc.) are extracted in order to use them as indices to key frames and hence to shots. Temporal features (e.g. object motion) can be used too.
- The retrieval phase: In this stage, a query is presented to the system that in turns performs similarity matching operations and returns similar data (if found) back to the user. One technique that is commonly used to present queries to video databases is QBE (Query By Example) [96]. In this technique, an image or a video clip is presented to the system and the user requests the system to retrieve similar items.

In this thesis, we present a new paradigm for solving the problem of content-based indexing and retrieval of video data. The proposed system tries to achieve its objectives by following a novel and effective approach that attempts to model the way humans perceive the similarity of video data.

In spite of the fact that a number of video indexing and retrieval prototype systems have been introduced by other researchers, we believe there are still essential problems that require better solutions. These solutions should aim at improving the reliability, efficiency, and effectiveness of video retrieval systems. The first shortcoming of most of the current video retrieval systems is the lack of reliability of the shot boundary detection stage [36]. The multiplicity of video streams, their varying contents, and the huge amounts of data involved are some obstacles against the design of robust and efficient techniques for detecting shot boundaries. Moreover, the lack of reliability of this particular stage not only affects its performance but also impacts the performance of the whole retrieval system. That is because the output of this stage plays a significant role in determining the results of all subsequent stages. The developed system introduces a novel paradigm to detect scene changes that is both reliable and efficient; thus, solving the problems exhibited in other shot boundary detection methodologies.

The second problem that will be addressed is how to devise an efficient algorithm to abstract the large amount of information found in each segmented shot. Most of the current approaches are either oversimplified so that they cannot perform the right choice or too complex that renders them unsuitable for on-line processing. Two efficient algorithms to select key frames are introduced with the goal of avoiding the aforementioned shortcomings.

Deriving content indexes from the selected key frames is the next stage in which we use two low-level features (color and texture) as the basic components of the generated metadata. These metadata will be used in any further processing or similarity matching operations. The effectiveness of the retrieval stage is the last issue we are concerned with. This problem is so critical in determining the success of a content-based retrieval system for video data. Currently, retrieval systems overlook a very essential fact while measuring the similarity of video data. That fact can be stated as similarity matching has significance only if it can model what humans do. On the contrary of other techniques, the proposed retrieval system introduces a new similarity matching approach that attempts to model the way humans perceive multimedia data and judge their similarity. The developed retrieval module handles a number of shortcomings in current prototypes; thus, improving the overall performance of the retrieval system.

1.1 Related Work

In this section, we review some related work that aims at solving the problem of CBR (Content-Based Retrieval) of video data, highlighting the challenges and the inadequacies of current approaches. Detailed literature reviews for each stage of the developed CBR system are provided in the relevant chapter. To start with, we first give a quick overview to the concept of indexing and retrieving digital images based on their contents [67] due to its relevance to the problem at hand.

The ideal way to describe the content of an image is in terms of its objects. However, object recognition in a general image database is a very hard problem. Instead, researchers extract low-level features (color, texture, shape, structure, spatial relationships among objects, etc.) to describe the content of an image. These features are then employed as indices to the image. Extracting these features and storing them into the database constitute the first stage of indexing images by content. The main functionality of the second stage, the retrieval system, is to analyze the presented query image in order to extract the same features from it. After that, the retrieval system performs similarity matching operations between extracted features of the query and those stored into the database. A number of systems have been proposed in the literature that, in general, apply the above-mentioned approach to access and browse images based on their contents [1], [2], [21], [31], [34], [44], [52], [61], [64], [72], [74], [76], [85].

On the other hand, by looking at indexing and retrieving of video data, the problem becomes much more complicated simply because video data have both temporal and spatial dimensions [38]. The research in this area can be broadly categorized into four main trends [65], video analysis, video representation, video browsing, and video retrieval. Video analysis can be further classified into shot boundary detection and key frames selection.

There are two approaches in the literature to segment video data. The first one works in the uncompressed domain while the other works in the compressed domain [13]. In the context of the first approach, some researchers use template matching techniques [35], [98], while others employ histogram based techniques [80]. A different group uses block-based methodologies [38] to detect shot boundaries. The major drawback of these techniques is their inefficiency because they work on uncompressed data. In compressed data domain. most of the introduced techniques utilize the DCT coefficients of compressed video to detect shot boundaries [14]. [94]. One common criticism to most of shot boundary detection techniques is the lack of efficiency and reliability: therefore. our system attempts to solve this issue by using a novel approach that is both efficient and reliable.

With respect to extracting representative frames. some researchers use only the first or the nth frame in a shot to represent it [58]. while others employ more complicated techniques (such as optical flow calculation) to extract key frames [90]. The first trend is so primitive. in most cases. and cannot faithfully represent the shot or capture its salient characteristics. The other one may produce more accurate results but it is computationally expensive that precludes its use in real-time retrieval systems.

Video structure is another area of research that has been investigated. Most of the successful techniques were proposed in the context of a specific application [50]. [101] where different models were proposed to abstract the video data based upon prior knowledge of that specific application domain. These techniques cannot be applied to general video databases that have various video genres.

Browsing techniques in the current literature can be broadly divided into three different directions. The first one of them simply displays key frames on the screen as a way of summarizing the video data. One example of such tools called the hierarchical video magnifier [75]. The second trend attempts to display the video as segments. A tool called clipmap proposed in [75] can be classified under this trend. It displays 3D icons called Micons (Movies icons) each of which represents a video shot where the third dimension is used to display the duration of the shot. The third direction is based upon the mosiacing technique [39]. Mosiacing is the representation of an entire shot as a single frame that includes all its static and dynamic information with no redundancy. A mosiac image is constructed from all the frames in a shot by aligning frames in a fixed coordinate system and using temporal filters to integrate them into a single image. Foreground objects are superimposed on this static background image.

The last stage in a video indexing and retrieval system is the retrieval phase. In this phase. the major commonly used technique to present the query is QBE [96]. which has the obvious advantage of expressing the query intuitively. Other models are also proposed to submit queries to video retrieval systems. One of them is introduced in [45] that uses query languages based on the semantic levels. Another research effort [4] proposes the use of 3D interfaces and virtual reality instead of using the QBE model. The major disadvantage of the last two techniques is that they are difficult to use and require experienced users to apply them effectively.

The central task in the retrieval subsystem is the similarity measure operation. There are many techniques to measure similarity but almost all of them are criticized in [70]. We address this problem by proposing a novel approach for measuring video data similarity. This approach is a central theme in this thesis and its details are given in CHAPTER V.

1.2 Objective

The main objective of this thesis is to devise a novel video content-based indexing and retrieval system whose main task is to endow its users with an easy-to-use, effective, and efficient scheme for retrieving the required information.

By adapting the QBE query model, the system allows users to issue their queries based on the content of the required video clip and avoids the need to formulate the query using keyword-based approach. A central notion of that system is its attempt to represent the content of multimedia data and measure their similarity in a way consistent with human perception. To implement this notion, we propose a set of new techniques. The first technique we propose to detect shot boundaries, the neural network paradigm, has already been inspired by the human biological system. This makes it a more appealing choice for solving a hard problem like the one we encounter. The following step is the application of two efficient algorithms to abstract each shot using a set of key frames representing the salient characteristics of that shot. The presented indexing scheme integrates two low-level features in a trial to effectively represent the video content.

One of our main contributions in this research is the proposed similarity measure technique. This technique used in the retrieval stage is based on the way humans perceive video data similarity [49]. To achieve this goal, the system attempts to model different factors involved in human judgment of video similarity. These factors include frames visual similarity, shots temporal order, rate of presenting video frames, and shot duration.

Our approach is based on modeling human perception meaning that. it attempts to use some factors that humans consider when judging video data similarity.

1.3 System's Components and Thesis Contributions

In this research. we address a number of subsystems constituting a fully content-based video indexing and retrieval system. In each of these subsystems. we present our contribution that solves the encountered problems while attempting to avoid the drawbacks of other approaches. These subsystems are:

- The shot boundary detection stage.
- The key frames selection module.
- The indexing techniques.
- The retrieval stage.

We introduce a brief overview description to each of these subsystems in this section. The remainder of this thesis is devoted to the description of the details of designing. implementing. and evaluating the performance of these four modules.

1.3.1 The Shot Boundary Detection Stage

Two basic challenges encounter the design of any shot boundary detection module. The first one of them is the tremendous amounts of data contained in video streams that necessitate very efficient algorithms to handle them for the detection module to be usable in real-time processing applications. The second challenge is the multiplicity of video types and the various characteristics and contents each of these types has. This challenge calls for reliable shot boundary detection techniques that can achieve good performance across a wide variety of video content. The performance should not degrade even in cases of very fast object motion or camera work that are found in many video genres such as action movies and racing video clips. We propose a novel paradigm in order to solve the problem of video scene change detection. This approach partly solves the lack of efficiency of other techniques by working directly upon compressed data. At first. an abstract form of the video stream. know as the DC sequence. is extracted directly from a compressed MPEG video. This sequence is used in an off-line phase to train a feedforward neural network in a supervised learning mode. The network learns the required mapping from the input space to the output space and stores it into its connection weights. These weights are then used in any further recall phase. The process of shot boundary detection is now a matter of extracting the DC sequence of the targeted video stream and using it as an input to the neural network during its recall phase. In that phase, the mapping information is retrieved from the network connection weights in order to detect shot boundaries in that video stream. The developed shot boundary detection technique achieves its superior efficient performance over its peers because of two factors:

- Working on abstract representation of the original video stream that dramatically reduces the amount of data to be processed.
- Using the instantaneous recall phase of the neural network to solve the problem of shot boundary detection.

Moreover, the novel use of the neural network as a shot boundary detection mechanism contributes to the outstanding reliability our system managed to achieve. By using that technique, we exploit the advantages of the powerful supervised learning concept in order to improve the reliability of the system. Once the neural network learns the mapping from the input space to the output space, it stores that mapping into its connection weights regardless how complex it is. We do not even need to mathematically formulate this mapping because this task has been delegated to the network and its learning algorithm. Other techniques use some heuristics or rules of thumbs to accomplish the detection task but most of them are not broad-spectrum enough to model the complex mapping incurred by various video types and characteristics. On the contrary, the developed technique is very flexible and extensible so that it can accommodate any required mapping extensions by retraining the network using representative samples of the new video types we need to deal with.

1.3.2 The Key Frames Selection Module

After segmenting a video stream into a group of shots, each of these shots still has a large number of frames. To effectively and efficiently index video data, the number of frames in each shot has to be reduced by selecting a representative subset of the shot's frames to represent it. This task, called key frames selection, relies on the inherent similarity between successive video frames as its basic operational principle.

Ideally, a key frame selection algorithm should select the minimum number of key frames required to faithfully represent all the salient characteristics of a video shot. To approach this ideal situation, the selection algorithm should maintain two properties:

- Efficiency: It should select the required set of key frames in a very short time.
- Accuracy: It has to select only the minimum and sufficient number of key frames required to represent the shot in a precise way.

Other approaches often lack one or even both of the above properties in their suggested solutions to the problem of key frames selection. Therefore, two adaptive algorithms are proposed to tackle the shortcomings of other approaches. Both algorithms achieve part of their efficiency by working upon the DC sequence extracted by the shot boundary detection stage.

The algorithms are adaptive in the sense that, they constantly change their selection decisions based on two factors:

- The dimension of the input video file.
- The amount of activity of each video shot.

The algorithms implement a first level of adaptation in order to accommodate various input video dimension without any impact on the selection performance. At the same time, both algorithms use a second level of adaptation to change the number of selected key frames for each shot based on how active this shot is. In other words, each algorithm detects the amount of activity in each shot then either increases the selection frequency for very active shots to accurately capture their salient characteristics or decreases the selection frequency of inactive shots to avoid choosing redundant frames. The second level of adaptation is the basic contributor to the accuracy property of the developed algorithms. On the other hand, the two factors contributing to the efficiency of the proposed algorithms are:

- The use of simple but efficient approaches.
- Working upon an abstract version of the original video data.

1.3.3 The Indexing Techniques

Indexing multimedia data is a broad subject with tremendous issues to be solved. For the purpose of our video indexing and retrieval system, we adapt two features extracted from the previously selected key frames to be used as metadata that describe the original video stream. These features are:

- Color histograms.
- Texture feature vectors.

We developed an efficient method to extract the color histograms of the selected key frames that works directly on the DC sequence. At first, color space conversion is performed then a color histogram is constructed from the RGB color space for each selected key frames. This color histogram is then used as one component of the feature vector representing that key frame.

The texture is extracted using the Gabor wavelet transform [53] due to its superior performance compared with other texture extraction techniques. The texture extraction algorithm works upon the original form of the video and not on its DC version in order to accurately capture the texture property regardless of the frequency of its patterns. The texture feature vector and the color histogram are the main constituent of the metadata describing each key frame. These metadata are then used in any further processing operations instead of the original video data.

1.3.4 The Retrieval Stage

This stage should provide the user with an easy-to-use interface through which the user can submit his/her queries and navigate the returned search results. In addition, the system should be efficient enough so that it can return the required information within an acceptable response time, the time between submitting the query and returning the search results. Moreover, the returned results should be relevant to the submitted query otherwise the usefulness of the retrieval system is questionable. To achieve all these objectives, a number of issues need to be addressed:

- The query model: What types of queries can the system accept?
- The scalability of the system: The impact of increasing the size of the video database on the response time.

- The system effectiveness: How similar are the returned results to the submitted query?

As mentioned before, the most famous model to submit a query to a video retrieval system is the QBE because of its advantages over keyword-based query model. Thus, the proposed system adapts the QBE model as a way to accept input queries.

The second important issue to be addressed is the system scalability. Coming up with a retrieval system that cannot scale with the increase of the video archive size is not that useful. To solve this problem, a two-stage search procedure is proposed. In the first stage, signatures of query shots (derived from their metadata) are compared with the signatures of database shots then relevant database shots are used as candidates to the next search stage. At the end of that filtering stage, almost all of irrelevant database video clips are excluded ending with a much reduced search space for the second stage. In the second stage, the accurate search one, the matching is performed upon the feature vectors then similar results are returned back to the user. The application of the filtering stage improves to a considerable extent the scalability of our retrieval system and facilitates handling large-size video archives. Moreover, it enhances the system response time experienced by users.

One of the most essential aspects of a video retrieval system is its effectiveness, meaning that, how close the returned results are to the submitted query. A central theme in improving the effectiveness of the system is the way it determines the similarity of video data. Stems from our belief in the importance of this process, we integrate a number of similarity measure factors that most likely humans employ to determine how two videos are similar to each other. In addition, we not only introduce these factors but also give the user the control to determine how important each of these factors is. This flexibility extends the use of the system to cover various situations and user preferences and objectives. This mechanism also exploits the powerful concept of computer-human interaction with the ultimate goal of improving the whole system performance.

One last issue we need to emphasize here is that all our software implementations are in Java [77]. This enables the use of the system across heterogeneous platforms, a real advantage over other video retrieval prototypes that use other implementation languages.

1.4 Outline

The rest of this dissertation is organized as follows. CHAPTER II introduces our novel paradigm for detecting shot boundaries in MPEG compressed video streams. The extraction of the DC sequence is discussed first then the design of the neural network module used to accomplish the detection task is described next. In CHAPTER III, two adaptive and efficient algorithms are developed to solve the problem of key frames selection. The adaptation strategies used in both algorithms are then discussed in detail highlighting the real need for them in order to improve the accuracy of the selection module. Two indexing techniques, color histograms and Gabor wavelet texture, are the focus of CHAPTER IV. This chapter discusses the implementation of the employed indexing techniques and evaluates their performance. CHAPTER V investigates the design and implementation of the proposed video retrieval stage. This stage is by far the most important stage as it determines the effectiveness of the whole video indexing system. We start by discussing the general architecture of the retrieval stage then examine the user interface and the browsing environment. After that, we introduce our novel similarity matching model and evaluate the performance of the whole system. Finally, conclusions and future work for this research effort are given in CHAPTER VI.

CHAPTER II

DETECTING SCENE CHANGES ON MPEG VIDEOS

The increasing use of multimedia streams nowadays necessitates the development of efficient and effective methodologies for manipulating databases storing this information. Moreover. content-based access to video data requires in its first stage to parse each video stream into its building blocks. The video stream consists of a number of shots each one of them is a sequence of frames pictured using a single camera. Switching from one camera to another indicates the transition from a shot to the next one. Therefore. the detection of these transitions. known as scene change or shot boundary detection. is the first step in any video analysis system. A number of proposed techniques for solving the problem of shot boundary detection exist but the major criticisms to them are their inefficiency and lack of reliability. The reliability of the scene change detection stage is a very significant issue because it is the first stage in any video retrieval system: thus. its performance has a direct impact on the performance of all other stages. On the other hand. efficiency is crucial too due to the voluminous amounts of information found in video streams.

This chapter proposes a new robust and efficient paradigm capable of detecting scene changes on compressed MPEG video data directly. At first. an abstract representation of the compressed video stream. known as the DC sequence. is extracted then it is used as input to a Neural Network Module that performs the shot boundary detection task. We have studied experimentally the performance of the proposed paradigm and have achieved higher shot boundary detection and lower false alarms rates compared to other techniques. Moreover. the efficiency of the system outperforms other approaches by several times. In short. the experimental results show the superior efficiency and robustness of the proposed system in detecting shot boundaries and flashlights. sudden lighting variation due to camera flash occurrences. within video shots.

2.1 Introduction

The recent explosive growth of digital video applications entails the generation of vast amount of video data. however. the technologies for organizing and searching video databases are still in their infancy. The first step in indexing video databases (to facilitate efficient access) is to analyze the stored video streams. Video analysis can be classified into two stages [65]. shot boundary detection and key frames extraction. The purpose of the first stage is to partition a video stream into a set of meaningful and manageable segments whereas the second stage aims to abstract each shot using one or more representative frames. We will address the problem of shot boundary detection in this chapter while the next chapter focus will be the problem of selecting key frames from segmented shots.

In general. successive frames in motion pictures bear great similarity among themselves but this generalization is not true at boundaries of shots. A frame at a boundary point of a shot differs in background and content from its successive frame that belongs to the next shot see Fig. 1. In a nutshell. two frames at a boundary point will differ significantly as a result of switching from one camera to another. and this is the basic principle that most automatic algorithms for detecting scene changes depend upon.

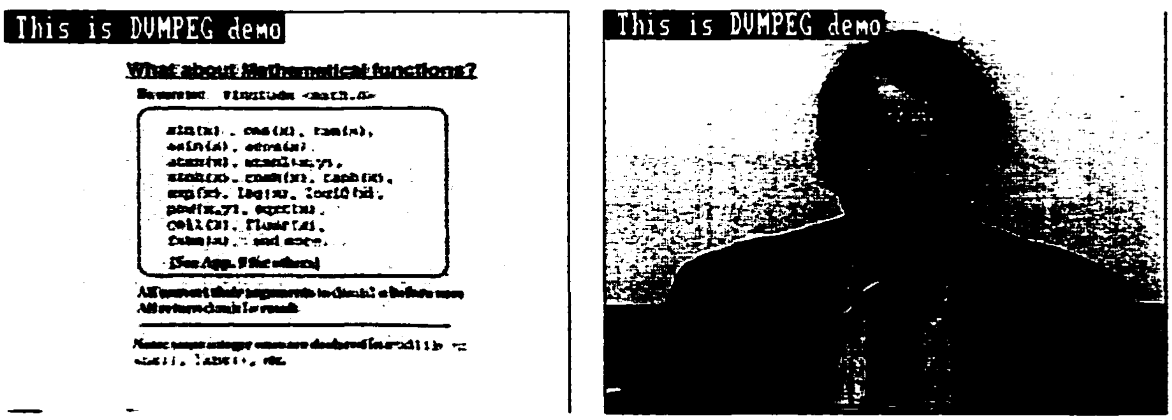

Fig. 1. The differences in content and background between two successive frames at a shot boundary.

Due to the huge amount of data contained in video streams. almost all of them are transmitted and stored in compressed format. While there are large numbers of algorithms for compressing digital video, the MPEG format [40], [47], [57] is the most famous one and the current international standard. In MPEG, spatial compression is achieved through the use of a DCT (Discrete Cosine Transform)-based algorithm similar to the one used in the JPEG standard [60], [88]. In this algorithm, each frame is divided into a number of blocks (8X8 pixel) then the DCT transformation is applied to these blocks. The produced coefficients are then quantized and entropy encoded, a technique that achieves the actual compression of the data. On the other side, temporal compression is accomplished using a motion compensation technique that depends on the similarity between successive frames on video streams. Basically, this technique codes the first picture of a video stream (I frame) without reference to neighboring frames while successive pictures (P or B frames) are generally coded as differences to that reference frame(s). Considering the large amount of processing power required in the manipulation of raw digital video, it becomes real advantage to work directly upon compressed data and avoid the need to decompress video streams before manipulating them.

In this chapter, a novel and robust algorithm for detecting shot boundaries is introduced [26]. The first module of the algorithm extracts an abstract description of a video frame that is known as the DC frame. DC, a term from Electrical Engineering that stands for Direct Current, is the first coefficient of the discrete cosine transform that is proportional to the average intensity of the block. A DC frame is generated by using the DC component of each DCT block and neglecting the AC coefficients, the other DCT terms. The DC coefficient has been chosen due to its central role. A series of these frames is called the DC sequence. This way, we managed to abstract the MPEG video stream by deriving its DC sequence without the need to decode the original stream. A modified version of the DC sequence is then used as input to a NNM (Neural Network Module) that performs the task of detecting shot boundaries found into that stream.

At first, we select some of the representative video clips from our database (with known shot boundary positions) as a training set and derive the DC sequences for them. These sequences are then merged to form one DC sequence to be used in training the neural network. The training is performed as an off-line activity until the network reaches an acceptable training error level. The result of the training process is a group of connection weights that store the appropriate mapping from the input space to the output space. These weights are stored to be used in the recall phase. Now, the process of detecting shot boundaries of a video clip is a matter of deriving the DC sequence of that clip. Then, using it as input to the network to recall the information stored into the connection weights during what is know as the recall phase of the neural network. That recall phase yields a set of shot boundary positions (if any).

The rest of this chapter is organized as follows. Section 2.2 briefly reviews a number of related approaches for detecting scene changes. The concept of representing the video stream using its DC sequence is introduced in Section 2.3 along with our algorithm for extracting that sequence. In Section 2.4, the design of the NNM that is used to detect scene cuts will be expounded. Experimental results are given in Section 2.5 followed by conclusion in Section 2.6.

2.2 Related Work

Video data are rich sources of information and in order to model these data, the information content of the data has to be analyzed. As mentioned before, video analysis is divided into two stages. The first stage is to divide the video sequence into a group of shots (shot boundary detection) while the second stage is the process of selecting key frame(s) to represent each shot. Generally speaking, there are two trends in the literature to segment video data. The first one works in the uncompressed domain while the other one works in the compressed domain. The first trend will be discussed first.

Methods in the uncompressed domain can be broadly classified into five categories, template-matching, histogram based, twin comparison, block-based, and model-based techniques. In template matching techniques [35], [98], each pixel at the spatial location (i,j) in frame f_m is compared with the pixel at the same location in frame f_n and a scene change is declared whenever the difference function exceeds a pre-specified threshold. Using this metric it becomes difficult to distinguish between a small change in a large area and a large change in a small area. Therefore, template-matching techniques are sensitive to noise, object motion and camera operations.

One example of the use of histogram-based techniques is presented in [80] where the histogram of a video frame and a difference function (S) between f_n and f_m are calculated. If S is greater than a threshold, a cut is declared. That technique uses equation (1) to calculate the difference function and declare a cut if the function is greater than a threshold.

$$S(f_m, f_n) = \sum_{i=1}^{N} \left| H(f_m, i) - H(f_n, i) \right| \qquad (1)$$

The rationale behind histogram-based approaches is that two frames that exhibit minor changes in the background and object content will also show insignificant variations in their intensity/color distributions. In addition, histograms are invariant to image rotation and change slowly under the variations of viewing angle, scale, and occlusion [78]. Hence, this technique is less sensitive to camera operations and object motion compared to template matching-based techniques.

Another technique that is called twin-comparison has been proposed in [98]. This technique uses two thresholds, one to detect cuts and the other to detect potential starting frames for gradual transitions. Unfortunately, this technique works upon uncompressed data and its inefficiency is the major disadvantage. A different trend to detect shot boundary is called block-based technique [38] that uses local attributes to reduce the effect of noise and camera flashes. In this trend each frame f_m is partitioned into a set of r blocks and rather than comparing a pair of frames, every sub-frame in f_m is compared with the corresponding sub-frame in f_n. The similarity between f_n and f_m are then measured. The last shot boundary detection technique working upon uncompressed data is termed model-based segmentation [38] where different edit types, such as cuts, translates, wipes, fades, and dissolves are modeled by mathematical functions. The essence here is not only identifying the transition but also the type of the transition.

On the other hand, methods for detecting shot boundaries that work in the compressed domain have been investigated. The main purpose of works in this trend is to increase efficiency. Again we can roughly divide these methodologies into three categories. The first category [14], [46], [94] uses DCT coefficients of video compression techniques (Motion JPEG, MPEG, and H.261) in the frequency domain. These coefficients relate to the spatial domain, hence they can be used for scene change detection. In [14] shot boundary detection is performed by first extracting a set of features from the DC frame. These features are placed in a high dimensional feature vector that is called the GT (Generalized Trace). The GT is then used in a binary regression tree to determine the probability that each frame is a shot boundary. Yeo and Liu [94] use the pixel differences of the luminance component of DC frames in MPEG sequences to detect shot boundaries. Lee et al. [46] derive binary edge maps from AC coefficients and measure edge orientation and strength using AC coefficients correlations then match frames based on these features.

The second category makes use of motion vectors. The idea is that motion vectors exhibit relatively continuous changes within a single camera shot while this continuity is disrupted between frames across different shots. Zhang et al. [99] have proposed a technique for cut detection using motion vectors in MPEG videos. This approach is based on counting the number of motion vectors M in predicted frames. In P-frames M is the number of motion vectors whereas in B-frames. M is the smaller of the counts of the forward and backward nonzero motion. Then. $M<T$ will be an effective indicator of a camera boundary before or after the B-and P-frames. where T is a threshold value close to zero.

The last category working into the compressed domain merges the above two trends and can be termed hybrid Motion/DCT. In these methods motion information and the DCT coefficients of the luminance component are used to segment the video [54].

Other approaches that cannot be categorized under any of the above two classes are reviewed below. Vasconcelos and Lippman [83] have modeled the time duration between two shot boundaries using a Bayesian model and the Weibull distribution then they derived a variable threshold to detect shot boundaries. A knowledge-based approach is proposed in [50]. [101] where anchorperson shots are found by examining intrashot temporal variation of frames. In order to increase the robustness of the shot boundary detection. Hanjalic and Zhang [36] proposed the use of statistical model to detect scene changes.

In summary. techniques that work upon uncompressed video data lack the necessary efficiency required for interactive processing. On the other hand. although the other techniques that deal directly with compressed data are more efficient. their lack of reliability is usually a common problem. To address these shortcomings. we proposed a reliable and very efficient technique to solve the problem of shot boundary detection of video data.

2.3 DC Sequence Extraction

Compressed MPEG files are the most commonly used forms for storage and transmission of audiovisual information. At the receiver side, the MPEG files need to be decoded in order to properly display the received information. In general, the decoding process is a highly time consuming task especially if it is done using software only and without any specialized hardware units. Consequently, working upon compressed data is a real advantage. The first step in any video retrieval system is to analyze the input files (MPEG files in our case) to detect shot boundaries. In order to achieve this objective while trying to enhance the efficiency of the proposed algorithm, our methodology attempts to detect shot boundaries directly from compressed data and avoids the requirement of decoding the video file first. This goal is achieved through two steps:

- Generating what is known as the DC sequence from the compressed data.
- Using the above-generated sequence in training a feedforward neural network that is used afterwards during its recall phase to detect shot boundaries.

2.3.1 Deriving Formulas for Extracting the DC Sequence

This section formulizes the problem of extracting the DC sequence from MPEG files and introduces our proposed solution to solve it. To encode MPEG files each frame in the original video file is divided into 8X8 blocks then the DCT transform is applied to individual blocks. The encoded transform coefficients in addition to the motion information are the main constituents of the compressed file. The first coefficient of the DCT of a block is termed the DC coefficient and it is directly proportional to the average intensity of that block. The main concept is to use these DC coefficients to derive an abstract description of a frame directly from the compressed data without the need for decoding. Each block will be represented by only one term (its average intensity derived from the DC term) and the composition of these terms will form what is called a DC frame. A sequence of such frames is termed the DC sequence. This sequence still bears high similarity to the original frame sequence [94] with the added advantage of that it can be directly and very efficiently derived from compressed data.

The general idea of using the DC sequence has been proposed in [71], [93]. The extraction of the DC frame from an I frame is trivial and can be calculated for each block as follows:

$$DC_I = \frac{1}{8} DC_{encoded} \quad (2)$$

Where

DC_I: The derived DC for a specific block.

$DC_{encoded}$: The encoded value in that block (the first coefficient of the DCT).

As Shown above deriving the DC terms from I pictures is a trivial task but for B and P pictures, it is not the same. One proposed solution in [71] is to calculate the DC of a block in B or P frames using equation (3).

$$DC_{P/B} = DC_{ref} + DC_{diff} \quad (3)$$

Where

DC_{diff}: The encoded DC coefficient of a block in a P or B frame.

DC_{ref}: The average of the DC coefficients of the reference frame blocks (at max four) overlapping with the predicted block.

Equation (4), given below, is the mathematical definition of the term DC_{ref}.

$$DC_{ref} = \frac{1}{64} \sum_{i=0}^{3} N_i DC_i \quad (4)$$

Where:

N_i : The intersecting area between the block in the P or B frame and the ith block in the reference frame, see Fig. 2.

DC_i: The DC coefficient of the ith block in the reference frame.

Note that, right horizontal displacement and downward vertical displacement are considered positive displacements in MPEG terminology [57]. Given the information in MPEG data and the proposed formulas described above, what is required is to determine two pieces of information in order to derive the DC sequence properly, these are:

- The intersecting areas of the blocks in the reference frame with the predicted block in the P or B frame. These areas are denoted N_0-N_3.
- The row and column indexes of each of the four intersecting blocks to be used in determining DC_i values, given the row and column indexes of the predicted block.

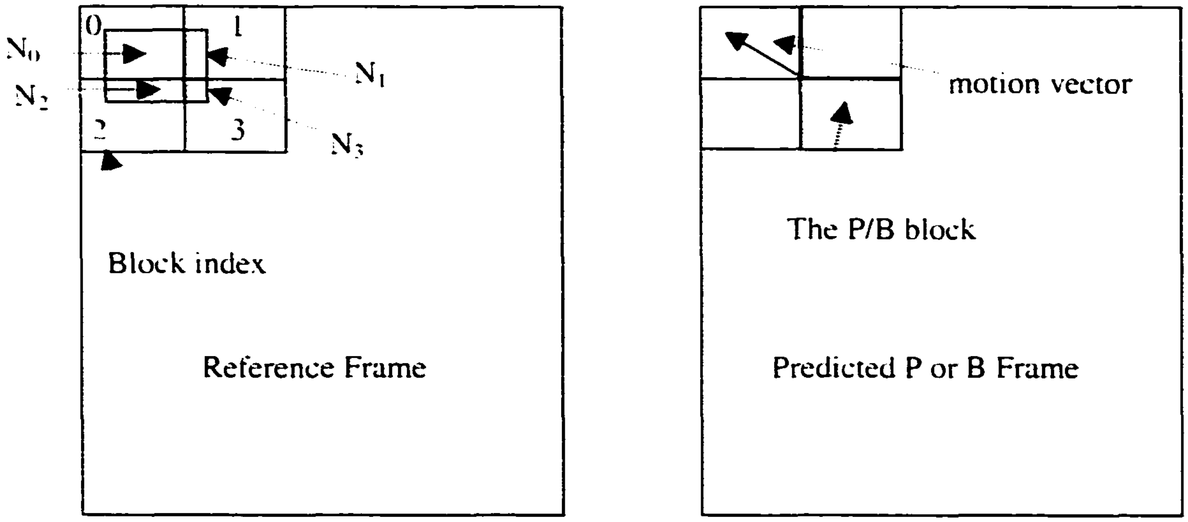

Fig. 2. The four intersecting blocks in the reference frame are to the left while the right shape shows the predicted block and the motion vector.

By analyzing the geometry of Fig. 2. the following formulas are derived to calculate the four areas. It is important to note that a different set of formulas is used for each combination of the signs of the motion vectors.

(1) Case of +ΔX and +ΔY (right horizontal and down vertical displacements)

$$N_0 = (L - |\Delta X|) * (L - |\Delta Y|)$$

$$N_1 = |\Delta X| * (L - |\Delta Y|)$$

$$N_2 = (L - |\Delta X|) * |\Delta Y|$$

$$N_3 = |\Delta X| * |\Delta Y|$$

(2) Case of +ΔX and -ΔY

$$N_0 = (L - |\Delta X|) * |\Delta Y|$$

$$N_1 = |\Delta X| * |\Delta Y|$$

$$N_2 = (L - |\Delta X|) * (L - |\Delta Y|)$$

$$N_3 = |\Delta X| * (L - |\Delta Y|)$$

(3) Case of -ΔX and +ΔY

$$N_0 = |\Delta X| * (L - |\Delta Y|)$$

$$N_1 = (L - |\Delta X|) * (L - |\Delta Y|)$$

$$N_2 = |\Delta X| * |\Delta Y|$$

$$N_3 = (L - |\Delta X|) * |\Delta Y|$$

(4) Case of -ΔX and -ΔY

$$N_0 = |\Delta X| * |\Delta Y|$$

$$N_1 = (L - |\Delta X|) * |\Delta Y|$$

$$N_2 = |\Delta X| * (L - |\Delta Y|)$$

$$N_3 = (L - |\Delta X|) * (L - |\Delta Y|)$$

Where

ΔX: The horizontal component of the motion vector.

ΔY: The vertical component of the motion vector.

L: The length of the side of a block (all blocks are squares and have the same dimension which is 8X8).

Three types of DC sequences one for each color components used in MPEG (Y, Cr, Cb) are extracted but we use only the Y component because human eyes are more sensitive to the luminance component than the chrominance ones [57]. If the block is bi-directionally predicted (has both forward and backward motion vectors), we propose to apply equation (4) to both the forward and backward cases and take the average value. This is a similar technique to how the MPEG algorithm handles the reconstruction of pixel values during the decoding phase. Thus in case of bi-directionally predicted blocks, equation (5) below will be used to evaluate DC_{ref}.

$$DC_{ref} = \frac{1}{2}\left(\frac{1}{64}\sum_{i=0}^{3} N_i DC_i(forward) + \frac{1}{64}\sum_{i=0}^{3} N_i DC_i(backward)\right) \qquad (5)$$

To specify which blocks in the reference frame will contribute to the DC_{ref} formulas, equations (4) and (5), we need to determine the rows and columns indexes of intersecting blocks in the reference picture and relates this information to the row and column index of the block under investigation. The influencing factors are the signs of the motion vectors. To derive the required relations, an investigation of the position of the considered block in relation to the other four intersecting blocks is performed. This investigation yields four sets of relations, one for each possible combination of motion vectors signs. These relations for rows and columns indexes of the four overlapping blocks (B_0-B_3) are listed in TABLE 1, assuming the row and column of the predicted block are R and C respectively.

By knowing the intersecting areas and the positions of the overlapping blocks in the reference frame, both equations (4) and (5) can be evaluated. One issue still needs consideration that is the case of large motion vectors, the focus of the next section.

TABLE 1

Rows and Columns Indexes for Overlapping Blocks as Functions of the Signs of Motion Vectors and the Position of the Predicted Block

	+ΔX and +ΔY		+ΔX and -ΔY		-ΔX and +ΔY		-ΔX and -ΔY	
	Row	Col	Row	Col	Row	Col	Row	Col
B_0	R	C	R-1	C	R	C-1	R-1	C-1
B_1	R	C+1	R-1	C+1	R	C	R-1	C
B_2	R+1	C	R	C	R+1	C-1	R	C-1
B_3	R+1	C+1	R	C+1	R+1	C	R	C

2.3.2 Handling the Case of Large Motion Vectors Magnitudes and Other Checks

The results in TABLE 1 assume that the magnitude of the MV (Motion Vector) cannot exceed the length of the block side (L), but in actual MPEG coded files this happens frequently and consequently the relations in TABLE 1 need to be adapted to account for such situations. To perform this adaptation (in case of magnitude (MV) > L), we calculate the value of a variable we called *addTerm*. This value will be added to the calculated rows and columns indexes in TABLE 1. In case of horizontal displacement, ΔX, the following algorithm is used to determine the value of the *addTermX*.

```
If( abs(ΔX) >= L ) {
    addTermX = ΔX/L
    ΔX = ΔX%L
}
else {
    addTermX = 0
```

$\Delta X = \Delta X$

}

The value of the *addTermX* will be added to all the calculated columns derived above in TABLE 1. The same procedure is applied in case of vertical displacement, ΔY, with the difference that its additional term, called *addTermY*, will be added to the calculated rows in TABLE 1. The final values of rows and columns of overlapping blocks are calculated below taking into account the case of large motion vectors magnitudes.

2.3.2.1 Determining Columns Indexes

To determine columns indexes, the following algorithm is employed.

If ($\Delta X > 0$) // *case of positive* ΔX

c0 = c2 = C+addTermX

c1 = c3 = C+1+addTermX

Else if ($\Delta X < 0$) //*case of negative* ΔX

c0 = c2 = C-1+addTermX

c1 = c3 = C+addTermX

Else // *case of* $\Delta X = 0$ *no horizontal displacement*

c0 = c2 = C+addTermX

2.3.2.2 Determining Rows Indexes

To determine rows indexes, the following algorithm is employed.

If ($\Delta Y > 0$) // *case of positive* ΔY

r0 = r1 = R+addTermY

r2 = r3 = R+1+addTermY

Else if ($\Delta Y < 0$) //*case of negative* ΔY

r0 = r1 = R-1+addTermY

r2 = r3 = R+addTermY

Else // *case of* $\Delta Y = 0$ *no vertical displacement*

r0 = r1 = R+addTermY

2.3.2.3 Special Cases

The following rules will be used to handle other special cases.

If $\Delta X = 0$ *Then* $N_1 = N_3 = 0$

If $\Delta Y = 0$ *Then* $N_2 = N_3 = 0$

If $\Delta X = 0$ *and* $\Delta Y = 0$ (the case of no motion compensation), then the DC will be calculated using equation (6).

$$DC_{P/B} = DC_{encoded} \qquad (6)$$

Where:

$DC_{encoded}$: The value of the DC of that predicted block embedded into the coding.

2.3.2.4 Boundary Condition Check

One important issue is to test the calculated values for rows and columns indexes to make sure that none of them is located outside the reference picture(s). To enforce this, the following checks (for luminance blocks) are performed for all the calculated values. Similar checks can be performed in case of chrominance blocks.

If(row < 0)

row = 0

*else if(row >= (mb_height * 2))*

*row = (mb_height * 2) -1*

if(col < 0)

col = 0

*else if(col >= (mb_width * 2))*

*col = (mb_width * 2) -1*

Where:

row: The calculated row index.

col: The calculated column index.

mb_width: The number of macroblocks in a row.

mb_height: The number of macroblocks in a column.

2.3.3 Handling the Case of Skipped MBs (Macroblocks)

The MPEG algorithm employs a smart way to encode macroblock that have all its DCT values equal to zero through the use of a macroblock increment value. The skipped macroblock can occurs only in P or B frames and cannot occur in I frames. One more condition is that the first and last macroblock in a slice has to be coded [57]. A skipped macroblock simply means that the value of the difference that is supposed to be coded is zero, so this area is exactly the same as its corresponding one in the reference picture. Our implementation of the DC extraction algorithm has to take this skipping into account, so the following steps are applied.

- If the frame is P frame, copy the DC values of the corresponding MB from the reference picture to the current position.
- If the frame is B frame and both forward and backward predictions are used, calculate the average of the DC values in both prediction frames then copy it as in the previous step.
- If the frame is B frame and either a forward or backward prediction is used, copy the DC from the corresponding block in the reference picture to the current position.

2.3.4 Handling the Case of non-coded Blocks

There is another situation that our algorithm has to take care of, that is the case of non-coded blocks specified in the CBP (Coded Block Pattern). The MPEG algorithm uses CBP to signify whether a block in a macroblock is coded or not. A block that is not coded means that all of its DCT coefficients are zeros. For coded blocks, the normal algorithm will be used to calculate the DC sequence. Otherwise, if the block is not coded our algorithm proceeds as follows:

- If the macroblock is Intracoded all the blocks in a MB should be coded, so no special procedure will be done.
- If the macroblock is Intercoded the values of the encoded DC values are assumed to be equal to zeros in any further calculation.

2.4 Detecting Shot Boundaries

The DC sequence extracted in the previous section is used as input to a NNM. The choice of the neural network [7], [105] as a shot-boundary detection methodology is based on its desirable generalization and fault tolerance properties. Detecting shot boundaries in a video stream is a hard task, especially when we consider the multiplicities of video types (action movie, romantic movies, sports, news cast, etc.) and the different characteristics that each of these types has. Many of the current shot boundary detection algorithms fail to detect shot boundaries in cases of fast camera or object motion or when a large object occupies the whole scene for a while. This lack of robustness in currently available techniques motivates us to propose a robust and efficient methodology to detect scene changes in various genres of video streams.

The first step in the design of the NNM is to determine a proper architecture of the network capable of solving the problem at hand [30]. Three architectures are investigated. in the first one shown in Fig. 3. the differences between corresponding DC terms in two adjacent DC frames are calculated and each difference value is used as input to a node at the input layer. Thus, for the jth element in the training/test set, the input to the input node i is given by equation (7).

$$Input_i(j) = DC_i(j) - DC_i(j+1) \tag{7}$$

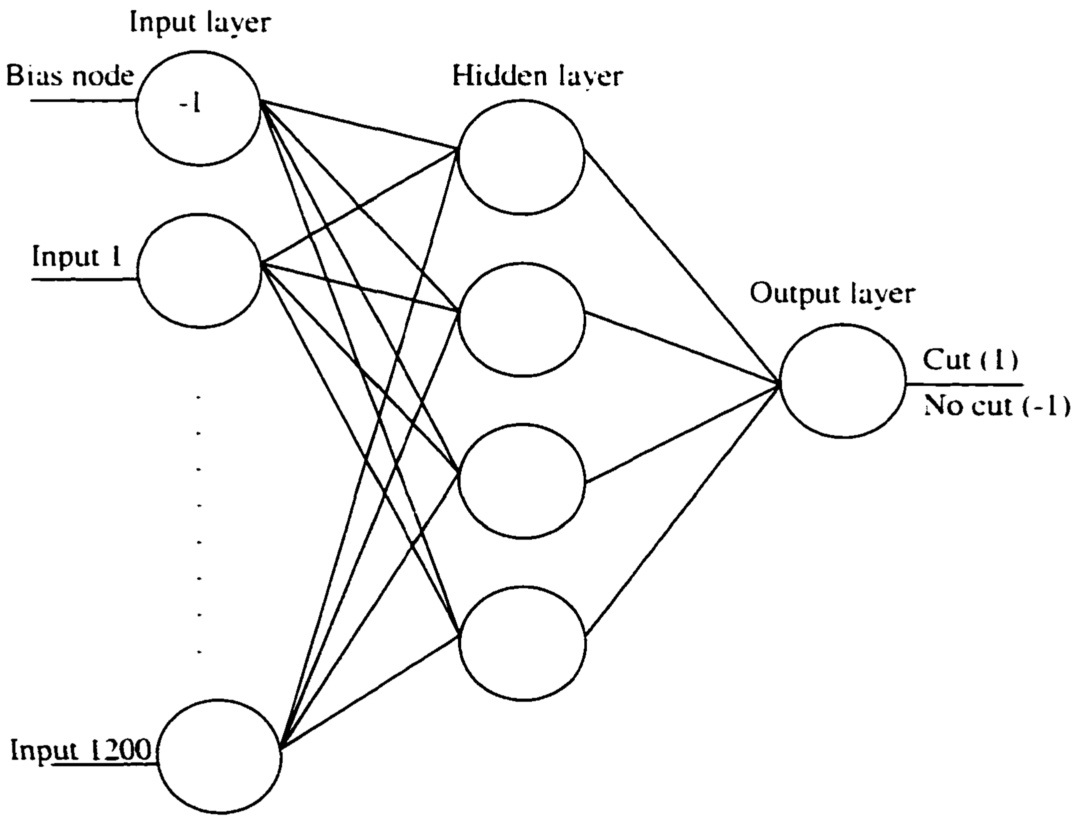

Fig. 3. The structure of the first proposed neural network for shot boundary detection. assuming an input MPEG video dimension of 320x240.

The second architecture diagrammed in Fig. 4 uses only one node (in addition to the bias node) at the input layer. The input to that node is the sum of absolute differences between corresponding DC values in two successive DC frames. Equation (8) defines the value of the neural network input for the jth element of the training/test set, where n is the number of DC terms in a DC frame.

$$Input(j) = \sum_{i=0}^{n-1} \left| DC_i(j) - DC_i(j+1) \right| \tag{8}$$

The last considered network structure illustrated in Fig. 5 employs three input nodes, each one of them accepts input as the previous architecture but for DC frame difference I (difference between j and j+1), I+1, and I+2 respectively. The inputs to this structure are formulated in equations (9), (10), and (11) respectively.

$$Input_1(j) = \sum_{i=0}^{n-1} \left| DC_i(j) - DC_i(j+1) \right| \tag{9}$$

$$Input_2(j) = \sum_{i=0}^{n-1} \left| DC_i(j+1) - DC_i(j+2) \right| \tag{10}$$

$$Input_3(j) = \sum_{i=0}^{n-1} \left| DC_i(j+2) - DC_i(j+3) \right| \tag{11}$$

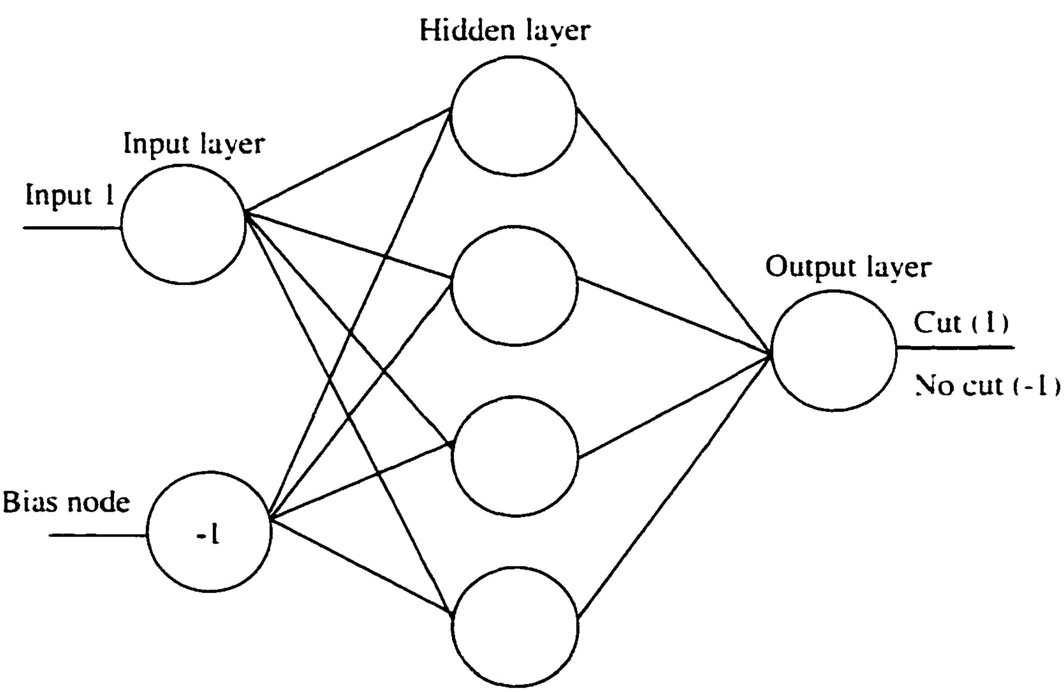

Fig. 4. The structure of the second proposed neural network for shot boundary detection.

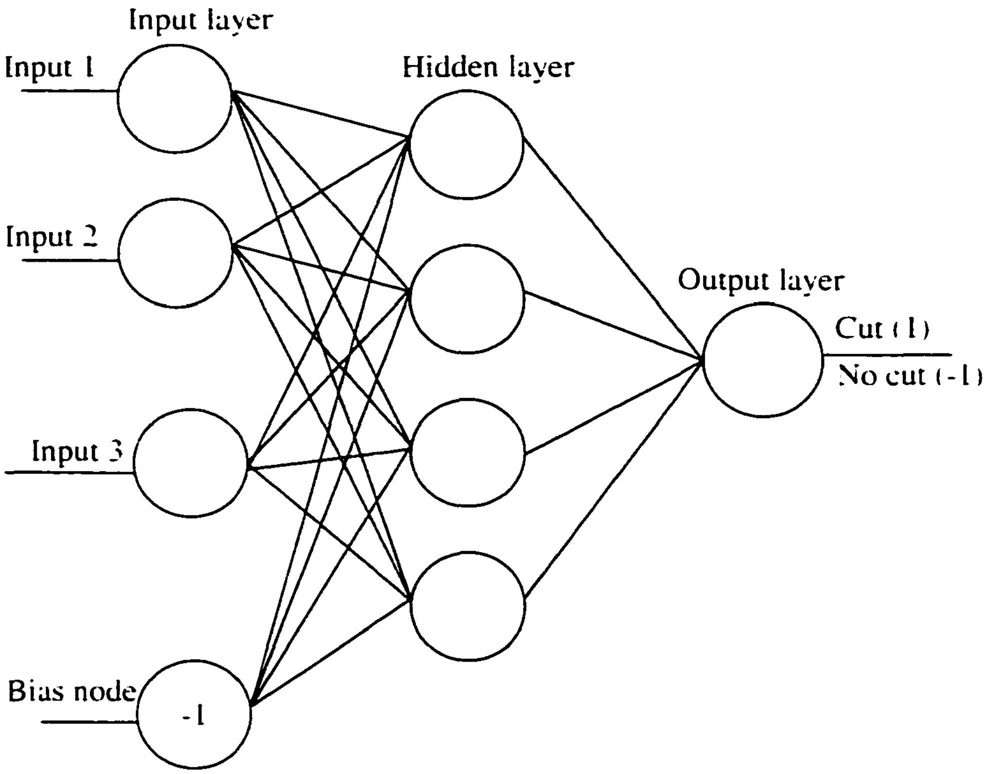

Fig. 5. The structure of the third proposed neural network for shot boundary detection.

The actual difference among the three architectures is the dimension of the pattern space presented to the network in order for it to learn the required classification task. In the first case the dimension of the pattern space is very large (it depends upon the input MPEG dimension), this implies complex network and longer training and recall time. In the other two architectures the input dimension is small (1 and 3 respectively). Our evaluation of these three architectures yields the following remarks.

- The dependence of the first architecture upon the dimension of the input MPEG clip results in presenting a difficult problem that has a very large input space dimension to the network.
- The large input space dimension entails the use of complex networks that require considerable amount of processing power to be trained or used in retrieving the embedded mapping information.
- Large and complex neural networks usually face difficulties to converge to acceptable training error levels.

- Even in cases where a complex network manages to converge, the performance of their recall phases is almost the same as the performance of other architectures.
- Both the second and the third structures are simple networks and there is no noticeable difference in the performance of their test phases.

Due to all of the above remarks, we opt to employ the second structure that is by far the simplest and most efficient one as will be illustrated in the next section. To train the neural network, we use a modified version of the back-propagation algorithm proposed in [68], this proposed algorithm works as follows:

- Determine the size of the network to be trained. This includes the number of hidden layers and the number of neurons in each layer.
- Determine the type of the activation function. In our case we use the bipolar sigmoid activation function defined in equation (12).

$$f(x) = \frac{2}{1+\exp(-\lambda x)} - 1 \qquad (12)$$

Where

λ: The slope, steepness coefficient, of the activation function.

- Determine the values of the training set and its size.
- Determine the values of the momentum back-propagation algorithm parameters. These are the learning rate (η) and the momentum coefficient (α).
- Randomly initialize all weights and thresholds.
- Present the first element of the training set to the network and initialize an error accumulator (E=0.0).
- Calculate the actual output of each neuron in all layers using equation (13).

$$y_{pj} = f\left(\sum_{i=1}^{n} w_{ji} x_i \right) \qquad (13)$$

Where

y_{pj}: Output of node j in any layer in response to presenting pattern p.

w_{ji}: The weight connecting node i in the previous layer to node j in this layer.

x_i: Output of node i in the previous layer.

f: The bipolar sigmoid activation function defined in equation (12).

n: The number of nodes in the previous layer.

- Use the responses of each layer as inputs to the next layer.
- Accumulate the error due to the actual output of each neuron in the output layer using equation (14).

$$E = E + \frac{1}{2}(d_{pk} - o_{pk})^2 \tag{14}$$

Where

d_{pk}: Desired output of node k at the output layer in response to input pattern p.

o_{pk}: Actual output of node k at the output layer in response to input pattern p.

- Calculate the error signal at any node j for a pattern p using equation (15) for the output layer and equation (16) for hidden layer(s).

$$\delta_{pj} = \frac{1}{2}\lambda(1 - o_{pj}^2)(d_{pj} - o_{pj}) \tag{15}$$

$$\delta_{pj} = \frac{1}{2}\lambda(1 - o_{pj}^2)\sum_{k=1}^{K}\delta_{pk}w_{kj} \tag{16}$$

Where

δ_{px}: The error signal at node x as a result of presenting pattern p.

w_{kj}: The weight connecting node j in this layer to node k in the next layer.

K: The number of the neurons in the next layer.

- Adapt each layer weights using equation (17).

$$w_{ji}(t+1) = w_{ji}(t) + \eta\delta_{pj}o_{pi} + \alpha\Delta w_{ji}(t) \tag{17}$$

Where

$\Delta w_{ji}(t)$: Wight adaptation at time t.

- For all of the remaining elements in the training set go and repeat execution starting from equation (13).
- If $E < Emax$, then store the connection weights then exit, otherwise proceed.
- If the number of iteration exceeds a maximum value stop declaring convergence failure, otherwise initialize the error accumulator ($E=0.0$) and repeat execution starting from equation (13).

To perform the recall phase of the network a similar algorithm to the one just described is used but without any weights adaptation. Instead, the resulting weights produced during the training phase will be used to calculate the output of the network in the feed forward direction via equation (13).

To determine proper values for the parameters of the back-propagation algorithm and the neural network, many combinations of different values have been tested in order to select those that give better results. These parameters include the number of hidden layers, number of neurons in each hidden layer, the learning rate, the momentum coefficient, the slope of the sigmoid function, and the number of nodes at the input and the output layers. TABLE 2 shows the best values obtained for these parameters out of our experimentations.

TABLE 2

Neural Network and Back-propagation Algorithm Parameters Used in Training and Testing Phases

# of input nodes	# of hidden layers	# of nodes in each hidden layer	K	η	α	λ
structure dependent	1	4	1	0.4-0.8	0.2-0.4	1.0

2.5 Experimental Results

At first, the network is trained using a combination of two video clips. The first clip is a soccer match video that has one cut while the second one is a wrestling clip that has two cuts. The network learned the classification task quickly and stored the mapping between the inputs and outputs into its connection weights. In spite of the small training set used, the convergence behavior of the network was very good as shown in Fig. 6 that depicts the learning error throughout the course of the training phase. The Y-axis of Fig. 6 has a logarithmic scale to illustrate the rapid decay in learning error as training cycle increases. The next step is to test the generalization of the network during the recall phase. Many video clips (about sixty) from various situations have been used in the testing phase. The results were very good in which the network was able to detect almost all cuts in these clips although they have never been presented to the network before. TABLE 3 shows shot boundary detection results for twelve clips from our database.

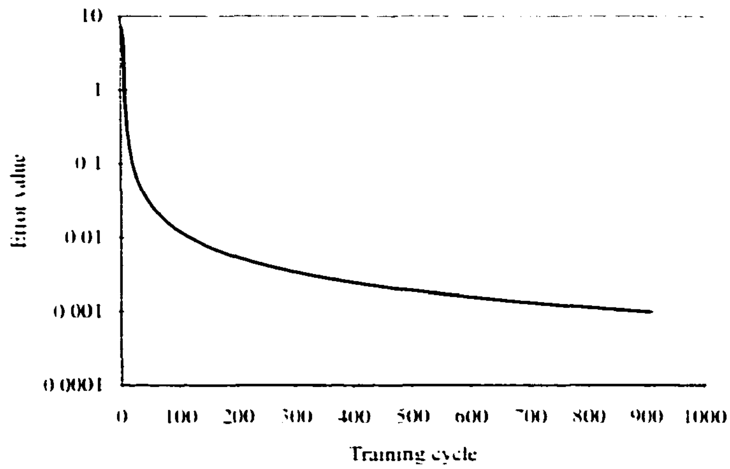

Fig. 6. The error value throughout the neural network training phase.

TABLE 3

Shot Boundary Detection Results for Various Video Genres

Video name	# of frames	# of cuts	# of detected cuts	False alarms
soccer	171	2	2	0
racing-boats	263	4	4	1
action-movie	178	2	2	0
carton	331	8	8	0
celebration	457	1-(5)	1-(4)	0
comedy	655	4	4	0
ads	759	10	10	0
class	2793	6	6	0
news-cast	2321	20	20	2
conf-discussion	4783	19	19	0
documentary	5094	41	35	2
tv-show	6139	72	72	0

At this point some analysis of the robustness of the obtained results is worthy. Consider the first clip in TABLE 3, the soccer video, where the frame difference graph for this clip is shown in Fig. 7. This clip has a dimension of 352x288 with 171 frames. There are two cuts at indexes 9 and 58, assuming the first frame difference index is 0. It can be observed from the graph that the two cuts are distinguished as two high peaks but at the same time there is a number of local peaks with comparable heights to the two cuts for example the one at 65. These local high peaks are results from fast object motion in addition to fast panning of the camera. Many algorithms that use threshold to detect shot boundary will be fooled by these FAs (False Alarms). Moreover, other algorithms that use different rules of thumb such as the cut peak should be twice as large as any surrounding peak [94] will miss the actual cut due to the presence of these local peaks. The robustness of our algorithm is evident in that particular clip where it detects only the two cuts and discards all false alarms.

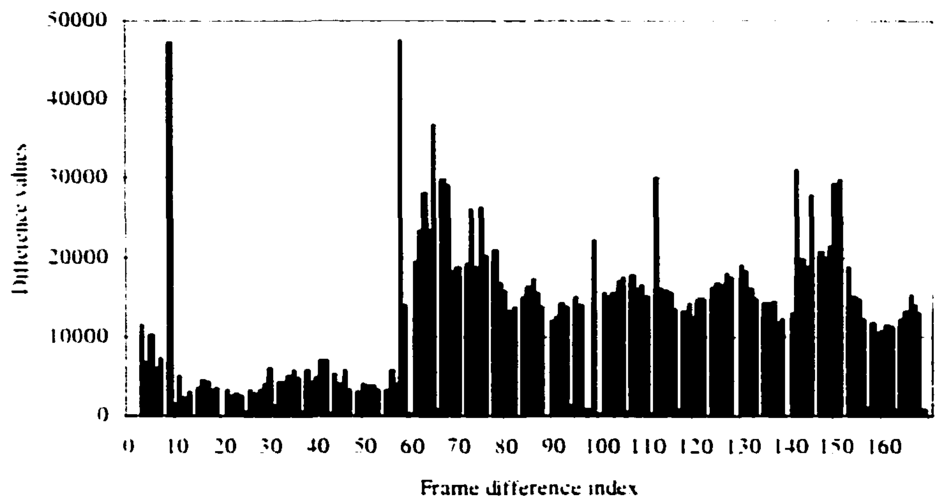

Fig. 7. The frame difference graph for the soccer match.

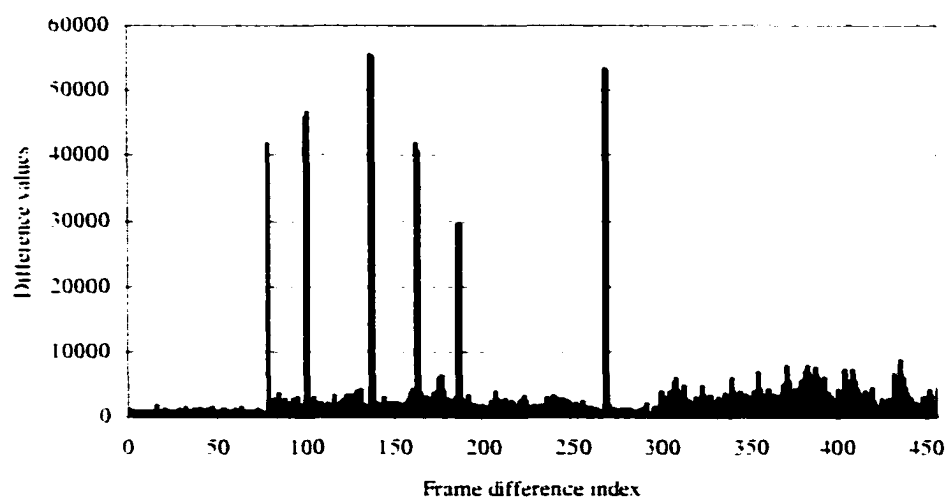

Fig. 8. The frame difference graph for the celebration clip.

Take from TABLE 3 the celebration clip as another example where Fig. 8 illustrates the frame difference for such a clip. This clip has a dimension of 320x240 with 457 frames. There is only one cut at 78, assuming the first frame index is 0. This cut is observable as the first high peak in Fig. 8, the frame difference diagram. There are also five high peaks in the diagram at (99, 137, 162, 187, and 268), actually each one of these peaks is two adjacent peaks, for instance there are almost two similar-value peaks at 99 and 100. Due to the large difference in lighting incurred by the occurrence of a flashlight, the frame difference diagram will have two consecutive large peaks of almost the same value for each occurrence. These peaks are a property in the frame difference diagram that indicates the presence of flashlights at these points; hence, we use this property to detect their occurrences.

The detection performance observed from investigating TABLE 3 is very good. All cuts have been detected except in the documentary clip that has a lot of lighting variations that causes some misses. FAs are minimum, they happen in the racing-boats clip where a large object occupies the whole scene and in the news-cast and documentary videos where dissolve-like transitions are incorrectly detected as cuts. Four flashlights out of five are detected in the celebration clip because of the weakness of the missed one. Based on the results in TABLE 3 the overall detection percent is almost 97% with 2.6% false alarms. These detection rates outperform the reported results of other proposed methods for detecting shot boundaries [36], [46]. Moreover, we use various types of videos and most of the used clips contain very fast camera work or object motion. Other algorithms fail under these fast motions; on the contrary, ours performs very well in all these situations.

A longer list of the segmentation results we obtained by applying the proposed algorithm to other video clips in our database is give in TABLE 4. These results support the effectiveness of the proposed technique and its generalization ability even when applied to wider range of video data with different contents and characteristics.

TABLE 4

Detailed Shot Boundary Detection Results for Various Video Clips

Video name	Dimension	# of frames	# of cuts (at positions)	# of detected cuts	False alarms
news-cast	160x120	2321	20	20 (same)	2
tv-show	160x120	6139	72	72 (same)	0
smg-npa-3	160x120	1115	16	16 (same)	0
srose-p2	160x120	3535	22	22 (same)	0
srose-p4	160x120	1678	10	10 (same)	0
class	176x112	2793	6 (442, 804, 1452, 1866, 2306, 2641)	6 (same)	0
adecco	176x112	273	0	0	0
conf-discussion	176x120	4783	19	19 (same)	0
enterprise	176x144	400	0	0	0
crawle	176x144	235	1 (87)	1 (same)	0
dbvath-qcif	176x144	179	1 (121)	1 (same)	0
hoey-v-kill	176x144	310	0	0	0
carton	304x224	331	8	8 (same)	0
v-hi	320x240	1800	2 (316, 987)	2 (same)	0
documentary	320x240	5094	41	35	2
tv-accident	320x240	5841	28	23	2
baby-f	320x240	245	0	0	0
tennis1	320x240	79	0	0	0
celebration	320x240	457	1 (5 flashes)	1 (4 flashes)	0
racing-boats	320x240	263	4 (25, 68, 142, 218)	4 (same)	1
mov0008	320x240	378	0	0	0
speed167	320x240	311	3 (151, 177, 292)	3 (same)	0
sprint	320x240	479	1 (210)	1 (same)	0
ah6a27	320x240	253	1 (68)	1 (same)	0
necktw	320x240	183	2 (46, 160)	2 (same)	0
sf-anna	320x240	1300	2 (314, 525)	2 (same)	0
corks	352x240	768	9 (68, 152, 179, 215, 462, 483, 516, 648, 675)	9 (same)	0
ads	352x240	759	10 (65, 97, 258, 407, 437, 579, 614, 650, 674)	10 (same)	0
action-movie	352x240	178	2 (51, 104)	2 (same)	0
close	352x240	751	0	0	0
hyakutake	352x240	526	0	0	0
fish	352x240	343	1 (296)	1 (same)	0
comedy	352x240	655	4 (48, 364, 500, 614)	4 (same)	0
lions	352x240	298	3 (205, 252, 278)	3 (same)	0
sq	352x240	255	0	0	0
039	352x240	1036	11	11 (same)	4 (large objects)
Adver-sound	352x260	123	3 (20, 43, 77)	3 (same)	0
04-eg4.mpg	352x288	171	2 (9, 58)	2 (same)	0
crawley	352x288	283	1 (95)	1 (same)	0
ah6a4b	352x288	181	1 (34)	1 (same)	0
fai2-3	352x288	253	1 (193)	1 (same)	0
dbvath_cif	352x288	175	1 (129)	1 (same)	0
hoey_v_kill_cif	352x288	377	1 (327)	1 (same)	0

To evaluate the efficiency of our system, we compare its performance with other systems proposed by different researchers. Lee et al. [46] compared the performance of three techniques to detect shot boundaries and they called them DC, FB, and PM respectively. The average frames/second achieved by each of these techniques (averaged over four video clips) is listed in TABLE 5. To achieve an accurate comparison with the data listed in TABLE 5, we tried our best to perform all our measurements under the same conditions such as (operating system, type of the processor, and processor speed). After that, we used our technique to measure the number of frames that can be processed in one seconds for a number of clips from our database and list the results in TABLE 6. Fig. 9 shows a comparison between the average efficiency of the three methods evaluated in [46] and the performance of our proposed technique where the efficiency of our system is obvious.

TABLE 5

Average Speed Comparison of Three Scene Change Detection Methods from [46]

Method name	Average frames/sec	# of clips used in the average
DC	10.9	4
FB	2.3	4
PM	11.1	4

TABLE 6

Detection Time and Frames per Second for Some Clips from our Database

Video name	# of frames	Detection time (sec)	# of frames/sec
soccer	171	7.6	22.4
racing-boats	263	6.7	39.3
action-movie	178	5.5	32.5
carton	331	5.1	64.8
celebration	457	15.3	29.9
comedy	655	18.7	35.1
ads	759	20.9	36.3
class	2793	15.5	180.3
news-cast	2321	10.6	219.3
conf-discussion	4783	23.6	202.6
documentary	5094	35.7	172.0
tv-show	6139	137.4	37.1

Analysis of the results in TABLE 6 indicates that, our system managed to achieve an average of about 89.3 frames/sec (averaged over 12 video clips containing about 24000 frames). This achieved speed is more than 8 times faster than the fastest method to detect shot boundary reported in TABLE 5 and the performance gab is evident too in Fig 9. Although our algorithm were implemented in Java [77] which is inherently much slower than other native programming languages, it achieves a dramatic speed up over the fastest method reported in the literature, a sound evident of the efficiency of the proposed technique. An important factor that contributes the major share to this efficiency is our novel use of the instantaneous recall phase of the neural network to accomplish the shot boundary detection task.

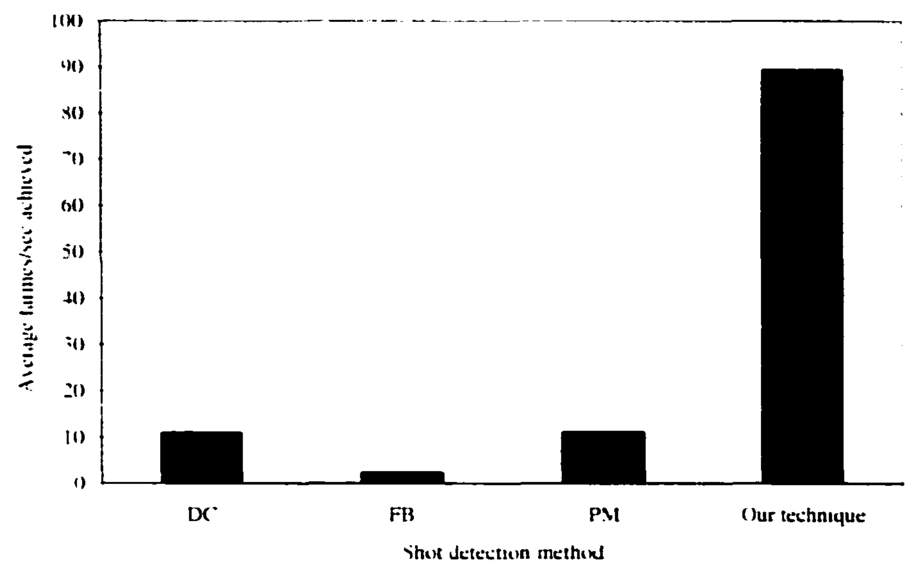

Fig. 9. Speed comparison of different shot detection techniques.

2.6 Conclusion

This work introduces an efficient and robust system for detecting video scene changes, an essential task in fully content analysis systems. The first module extracts the DC sequence directly from compressed data without the need for decoding. Subsequently, the second module receives DC frame differences as inputs then recalls the information stored into the neural network weights to determine the outputs. The algorithm has been tested on wide varieties of video genres and proved its robustness in almost all cases where detection percent of 97% was achieved with 2.6% FAs. Better generalization of the neural network can be achieved by increasing the number of video clips used in the training phase and by varying their contents. Networks that generalize better provide superior performance and widen the applicability of our technique to cover every possible video genre. Moreover, the efficiency of the developed technique has been tested by measuring the number of frames per second that the system can process. The test yields an average value of about 89.3 frames/second. This average value represents a dramatic speed up in comparison with the speed of other systems that perform the same shot boundary detection task. In a nutshell, the effectiveness of the proposed paradigm has been proven as a robust and very efficient way to identify scene changes in compressed MPEG video streams. The proposed video detection paradigm introduced in this chapter is the first stage in our content-based retrieval system for video data.

CHAPTER III

ADAPTIVE SELECTION OF KEY FRAMES

As multimedia applications are rapidly spread at an ever-increasing rate. the call for novel techniques for organizing and abstracting the data they produce becomes a necessity. One of the basic problems encountering such systems is to find efficient ways to summarize the huge amount of data involved. To solve this problem. video streams need to be analyzed first by dividing each stream into a set of meaningful and manageable units. a task that was the focus of CHAPTER II. After dividing a video sequence into a number of shots. each shot still contains a large number of frames. As a result of that. the second stage in any video analysis system is the process of KFs (Key Frames) selection [65] that aims to abstract the whole shot using one frame or more. Ideally. we need to select the minimal set of KFs that can faithfully represent each shot. KFs are the most important frames in a shot since they may be used to represent the shot in the browsing system as well as be used as access points. Moreover. one advantage of representing each shot by a set of frames is the reduction in the computation burden required by any content analysis system to perform similarity matching on a frame-by-frame basis. as we will discuss in CHAPTER V.

In CHAPTER II. we have explained how an input MPEG stream is processed first by extracting the DC sequence from it. Subsequently. that sequence is used as an input to a neural network module to perform the segmentation. shot boundary detection. task. The output of the segmentation stage is a group of distinct shots with clear marks to the beginning and end of each segment. This output is then fed to our KFs selection module [27] which is the focus of this chapter.

We start by introducing a fairly simple and efficient algorithm that uses the accumulated summation of DC frames luminance differences in order to select KFs from segmented video shots. Once the accumulated summation exceeds a certain threshold the current frame is added to the representative set in addition to the first frame (which is chosen by default) then the algorithm proceeds. To improve the basic algorithm introduced above. we propose a first-level adaptation mechanism that is based upon the dimension of the input video file. After that, a second level of adaptation based on a shot activity criterion is introduced to further improve the performance and accuracy of the selection module. At the end, another algorithm is proposed that uses a direct comparison between the summation of the luminance channel DC terms of the current frame and the corresponding summation of the last chosen key frame. The algorithm then selects the current frame into the representative set if the absolute difference is above a certain threshold. This algorithm uses a similar first-level adaptation policy used by the first group but it employs a statistical criterion for the shot-by-shot adaptation level. Analyzing the results produced by both algorithms showed their efficiency and accuracy in selecting the near optimal set of key frames required to represent each shot in a video stream. We then conclude the chapter by comparing the performance of both algorithms.

This chapter is organized as follows. We review briefly in Section 3.1 a number of related approaches for selecting KFs. The first proposed set of algorithms is then introduced in Section 3.2 starting by the most straightforward one and going towards the more effective ones. After that, Section 3.3 introduces the second proposed algorithm along with its criteria for the adaptation processes. Performance comparisons of the two proposed mechanisms are given in Section 3.4 followed by conclusion in Section 3.5.

3.1 Related Work

Key frames extraction is one of the active areas of research in visual information retrieval [9], [48]. A review of the major approaches that have been proposed by different researchers in the field to tackle this problem is given below.

A shot boundary based approach was proposed in [58] that uses the *nth* frame in each shot as the key frame. The main disadvantages are, this method uses only one key frame that may not be stable and may not capture the major visual content of the shot. Zhang et al. [97] proposed a visual content based approach where the first frame is used as a key frame but in addition to it other frames could be selected as additional key frames if they have significant content change compared to the first one. Motion based criteria are also considered in this method. Another approach that is based on motion analysis was proposed in [90]. That technique calculates the optical flow [73] for each frame, then computes a motion metric. It finally analyzes that metric and selects KFs at the local minima of motion.

A clustering algorithm has been proposed in [104]. The algorithm assumes that there are N frames within a shot divided into M clusters. The similarity between frames is performed using color histograms and the algorithm works as follows. The first frame is considered as the centriod of the first cluster then subsequent frames are compared with existing clusters centriods (if any). If the maximum similarity value is less than a threshold (δ) the current frame is put into a new cluster. otherwise it is put into the cluster with the maximum similarity. Finally. the algorithm adjusts the clusters centriods. After the process of clusters formation described above. the algorithm chooses KFs from representative clusters (those having the number of frames greater than N/M). The frame that is closest to the cluster centriod is chosen as a key frame.

In [95]. temporal sampling. selection of representative frames. is achieved by using a nonlinear sampling process in which every frame is compared with the last chosen representative frame. If the difference is above a certain threshold that frame is added to the set of representative frames for that particular shot. An alternative method is given in [94] where a set of KFs is used to represent a shot and the first frame in that shot is always selected as a KF. A different approach to represent the shot is proposed in [14]. where frames in the shot are organized in a tree structure in such a way that the root node is the most representative frame in the shot. As one progresses down the tree. frames are organized into representative groups. This tree representation is obtained through agglomerative (bottom-up) clustering. where color. texture. and edge histograms are the components of the feature vector and L1 norm is used to measure the feature distance.

Tonomura et al. [81] proposed a system that represents video sequences by using evenly spaced key frames while ignoring shot boundaries. The major problem with this system is that it selects more than the necessary number of key frames especially in long inactive shots. In [32]. a technique is introduced to detect key frames to represent the whole video without doing any shot boundary detection too. The general idea is to use a clustering algorithm to divide the frames into a number of clusters each one has similar frames then choose a frame from each cluster. The algorithm works as follows. At first. candidate frames are selected then the clusters are formed. Clusters that have small numbers of frames are filtered out (by using a time constraint condition) then another time constraint is used to guide the process of selecting key frames from clusters. At the end, some classes of images are emphasized as key frames, for example close-ups on people is preferred over long shots.

An illumination invariant approach is proposed in [23] to select key frames. The technique starts by an off-line processing stage that uses a training video set. It then normalizes the color channels, calculates the spherical chromaticity, applies the wavelet compression to the histogram produced by the previous step then uses the DCT transform to produce 21 DC coefficients. Finally, it employs singular-value decomposition to produce 12 basis vectors. For any new video the following on-line processing steps are performed. Process each frame as described in the off-line processing stage to produce 21 DCT coefficients that are used with the 12 basis vectors to provide 12-coefficient features vectors for that frame. Hierarchical clustering is then followed and frames closest to clusters centriods are selected as key frames.

A hierarchical color and motion segmentation scheme that is based on a multi-resolution implementation of the recursive shortest spanning tree is proposed in [5]. All segment features produced by that hierarchical segmentation are then collected together using a fussy multidimensional histogram to reduce putting similar segments into different classes. Afterwards, the extraction of key frames is performed for each shot in a content-based rate-sampling approach.

3.2 Accumulated Frames Summation Group of Algorithms

The produced shots from the shot boundary detection module designed in CHAPTER II need to be summarized by extracting KFs from them. Although a number of KFs extraction techniques were proposed in the literature as discussed in the previous section, they have the following shortcomings.

- Some of the proposed methodologies for KFs extraction use only one frame to represent a shot. In many cases one frame is not sufficient to semantically represent the shot specially if the shot is complex or contains a lot of motion.

- A different group employs algorithms that are oversimplified so that they cannot adapt to different changing situations: for instance, various input frame sizes and various activity levels within shots.
- Other techniques use mechanisms that calculate the optical flow or other complex models to select KFs. Although they may give more accurate results than simpler approaches, they are computationally expensive that renders them unsuitable for on-line processing.

To avoid these shortcomings, we propose mechanisms that attempt to take a balanced approach between the two extremes. They are not oversimplified like some of the surveyed techniques while they attempt to be more accurate as well as computationally efficient. As mentioned before, the use of one key frame is, in general, not sufficient to represent a shot unless that shot is a completely still one. As a result of this fact, our proposed approaches use a set of frames to represent each shot. This set may contain only one frame if that frame is capable of representing the salient characteristics of the shot.

In this section, we present a group of algorithms to select KFs that uses the AFS (Accumulated Frames Summation) of DC terms. We start by explaining the first version of the proposed algorithm, illustrating its performance and commenting on its shortcomings. Subsequently, two enhancements to the basic algorithm are introduced along with the experimental results obtained by implementing each enhancement. All the proposed algorithms in this section and in the next one work on the DC terms of the luminance channel of DC frames extracted as described in CHAPTER II.

3.2.1 Using Accumulated Summation without any Adaptation

The first developed algorithm to select KFs could be described as follows:

- *Initialize the representative set to be empty and select the first frame as the first element in that set.*
- *Initialize sum = 0 and flashIndex to the last detected flash position plus one.*
- *For j = 0 To N-2 do {*

  if (j = flashArray[flashIndex]) / only if ((j-1) >= 0) */*

  sum = sum + diffArray[j-1]

  else if (j = (flashArray[flashIndex]+1)) *sum = sum + diffArray[j-2]*

flashIndex = flashIndex+1

else

sum = sum + diffArray[j]

if (sum > δ_1)

sum = 0

Add j to the set of selected KFs

Increment the number of selected KFs

}

Where

$$diffArray[j] = \sum_{i=0}^{M-1} \left| DC_i(j) - DC_i(j+1) \right| \tag{18}$$

N: The number of frames in a shot.

flashArray[ ]: An array containing indexes to flashlight positions.

δ_1: An initial threshold that determines the frequency of sampling.

DC_i: The DC of the luminance channel at location i.

M: The number of DC terms in a DC frame.

It is important to note that the occurrences of flashlights within shots are detected by the shot segmentation algorithm and those points are passed to the key frames selection module into *flashArray[ ]*. The KFs selection module employs this information to avoid the inclusion of these pseudo peaks (see Fig. 8 in CHAPTER II) into the calculation of the accumulated summation used by the above algorithm. Instead, the algorithm substitutes each of these high-valued peaks by the last considered frame difference value (if any) as an approximation. In that way, the algorithm robustly excludes the effect of lighting changes occurred as results of flashlights.

In order to evaluate the performance of the above algorithm a number of video files were used during the tests. These files were segmented using the methodology proposed in CHAPTER II and a summary of the segmentation results was given in TABLE 3. Moreover, in our choice of these clips, we attempted to select various types of contents, amount of activities, and sizes of the captured video in order to prove the applicability of the proposed algorithms over a range of various circumstances. The next important issue that needs to be addressed in order for the above algorithm to be effective is the choice of the threshold value (δ_i) that gives best results for all these clips. While choosing the appropriate value of the threshold and in evaluating the proposed algorithms, we depend upon the human opinion in comparing the selected KFs with those that should be generated (ground truth). To achieve these goals, we start by trying various values of the threshold with almost all of the clips in TABLE 3 and investigate which threshold value gives the best results in terms of the minimal expressive set of key frames. TABLE 7 and TABLE 8 show the numbers of selected KFs, their averages per shot, and the total average as a function of threshold value for 11 clips from TABLE 3.

TABLE 7

Number of Selected KFs as a Function of Threshold Values ($25000 <= \delta_i <= 100000$)

Video name	δ_i = 25000		δ_i = 50000		δ_i = 100000	
	# of KFs	Ave / shot	# of KFs	Ave / shot	# of KFs	Ave / shot
soccer	56	18.7	31	10.3	18	6.0
racing-boats	37	6.2	22	3.7	14	2.3
action-movie	20	6.7	11	3.7	6	2.0
carton	50	5.6	29	3.2	18	2.0
celebration	42	21.0	23	11.5	15	7.5
comedy	46	9.2	25	5.0	13	2.6
ads	79	7.2	44	4.0	24	2.2
class	27	3.9	14	2.0	10	1.4
news-cast	71	3.1	43	1.9	27	1.2
conf-discussion	93	4.7	52	2.6	31	1.6
tv-show	167	2.3	107	1.5	81	1.1
Total Average		**8.03**		**4.48**		**2.72**

TABLE 8

Number of Selected KFs as a Function of Threshold Values ($150000 <= \delta_i <= 250000$)

Video name	δ_i = 150000		δ_i = 200000		δ_i = 250000	
	# of KFs	Ave / shot	# of KFs	Ave / shot	# of KFs	Ave / shot
soccer	13	4.3	10	3.3	8	2.7
racing-boats	9	1.5	8	1.3	8	1.3
action-movie	5	1.7	4	1.3	3	1.0
carton	15	1.7	12	1.3	11	1.2
celebration	8	4.0	7	3.5	6	3.0
comedy	10	2.0	8	1.6	7	1.4
ads	19	1.7	15	1.4	14	1.3
class	8	1.1	8	1.1	7	1.0
news-cast	25	1.1	24	1.0	23	1.0
conf-discussion	26	1.3	24	1.2	23	1.2
tv-show	75	1.0	74	1.0	73	1.0
Total Average		**1.95**		**1.65**		**1.64**

The relation between threshold value and the percentage of selected key frames to the total number of frames is plotted in Fig. 10 for three clips. The general and intuitive trend is the decrease in the number of selected key frames with the increase of the threshold values. This behavior is obvious for all the three curves in Fig. 10 and can be observed too by investigating TABLE 7 and TABLE 8. Our objective is to select the minimal number of key frames that is able to accurately describe and summarize the whole video stream. Selecting very large threshold will certainly reduce the number of selected KFs but at the same time will discard many necessary frames that are supposed to be selected in order to properly summarize the video data. On the other hand, small threshold values will produce many redundant key frames and render the similarity matching operation very inefficient. Bearing in mind that trade-off, we attempt to select a moderate value of the threshold that gives us the best results in terms of representation accuracy and the number of selected KFs. The value that balances this compromise is 150,000 and for that reason we will use this value in any further experimentations.

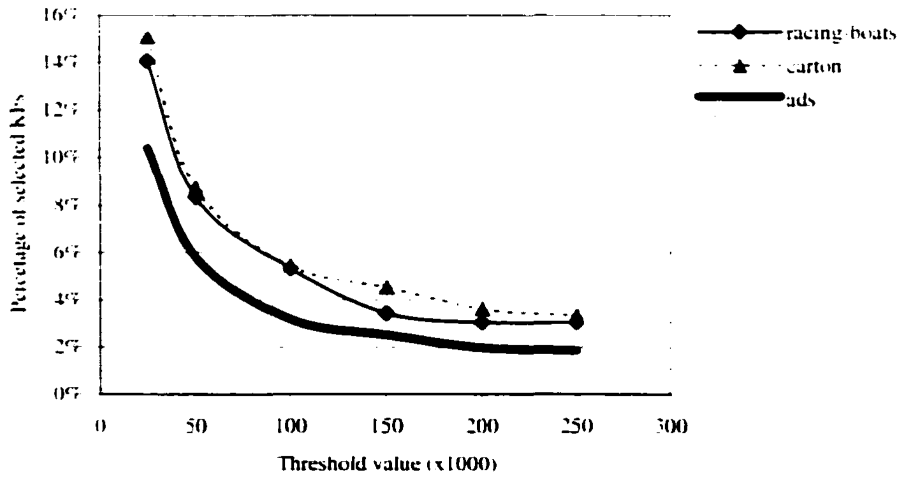

Fig. 10. The percentage of selected KFs as a function of the threshold value.

After selecting the value of the threshold that gives the best results, let us study in details the behavior of the algorithm when applied to a representative subset of the clips under investigation. Results for some long clips will be given in APPENDIX A. KFs selection results for the action-movie clip are shown in TABLE 9. Shots 0 and 1 are more active than shot 2 (check Fig. 11) and thus the algorithm chooses more KFs to represent these shots although the last shot has more frames than any of the other two. The basic factor to determine how many KFs needed to abstract a shot is the amount of activity in that shot but the shot length is another factor induced by the way the algorithm functions. It is important to note that, if the algorithm chooses three KFs for shot 0 they would abstract that shot better than only two frames as shown in Fig. 12.

TABLE 9

KFs Selection for the action-movie Clip Using δ_f =150,000 without any Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-51	52	2	0, 40
1	52-104	53	2	52, 96
2	105-177	73	1	105

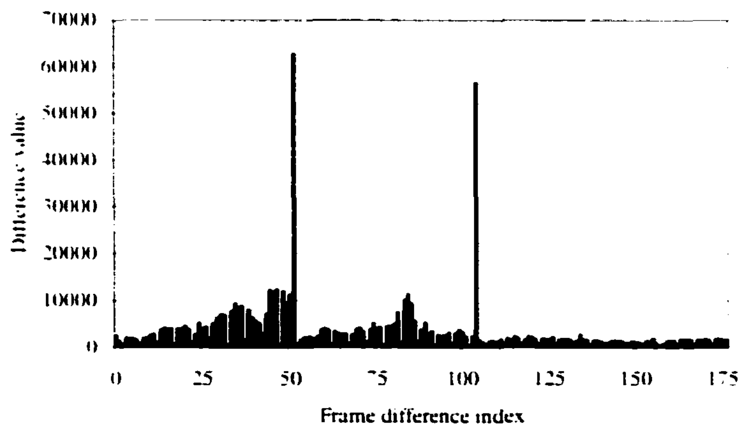

Fig. 11. The frame difference graph for the action-movie video.

Fig. 12. Three key frames better abstract the first shot of the movie clip than the use of two KFs because of the various positions the character takes.

For the carton clip, the moderate activity of shots 1 and 2 in TABLE 10 allows the algorithm to select two KFs for each one of them. Shot 3 has a considerable amount of activity that can be noticed by investigating Fig. 13 but the algorithm chooses only one KF to represent it. One more remark, the algorithm chooses the largest number of KFs for shot 4 that is actually the most active shot in this clip but regardless of that fact there is still a need for more KFs to properly represent that shot due to its high activity.

TABLE 10

KFs Selection for the carton Clip Using δ_f =150.000 without any Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-48	49	1	0
1	49-83	35	2	49, 80
2	84-133	50	2	84, 127
3	134-151	18	1	134
4	152-234	83	5	152, 164, 182, 203, 220
5	235-244	10	1	235
6	245-254	10	1	245
7	255-266	12	1	255
8	267-330	64	1	267

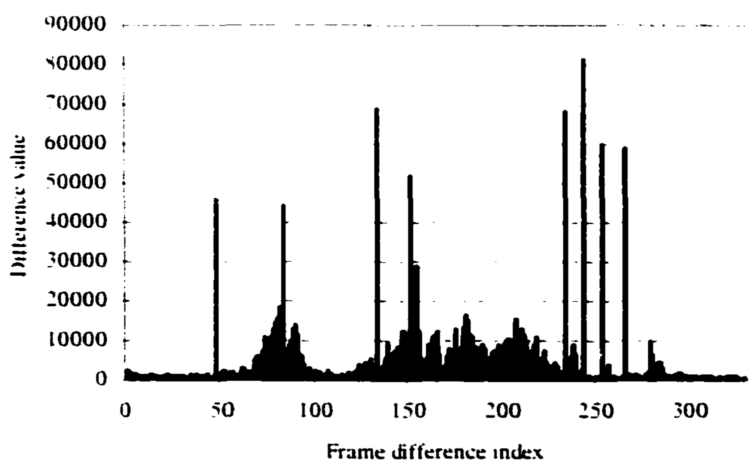

Fig. 13. The frame difference graph for the carton video.

Generally in TABLE 11, the result for the comedy clip whose frame difference graph is given in Fig. 14, one KF is selected every about 115 frames for low activity shots like shot 1 (as in the case between 49 and 163). This can change due to a local increase in the activity and reach about 60 frames (163-226) because of the entrance of a new character into the scene, see the second picture in Fig. 15. Ideally, three KFs are needed to properly describe shot 1, one should contains the two characters in the scene, another KF should reflect the entrance of a third character, and the last one should feature the presence of only one character at the end of that shot, see Fig. 15. The algorithm chooses 4 KFs for shot 1 as a result of its length. In shot 3, the selection of frame 581 as a KF in addition to frame 501 is justified by the entrance of a new character into the scene. The low activity nature of shots 0 and 4 can be noticed in the Fig. 14, thus the selection of one KF is proper. The choice for shot 2 is also appropriate.

TABLE 11

KFs Selection for the comedy Clip Using δ_f =150,000 without any Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-48	49	1	0
1	49-364	316	4	49, 163, 226, 331
2	365-500	136	2	365, 448
3	501-614	114	2	501, 581
4	615-654	40	1	615

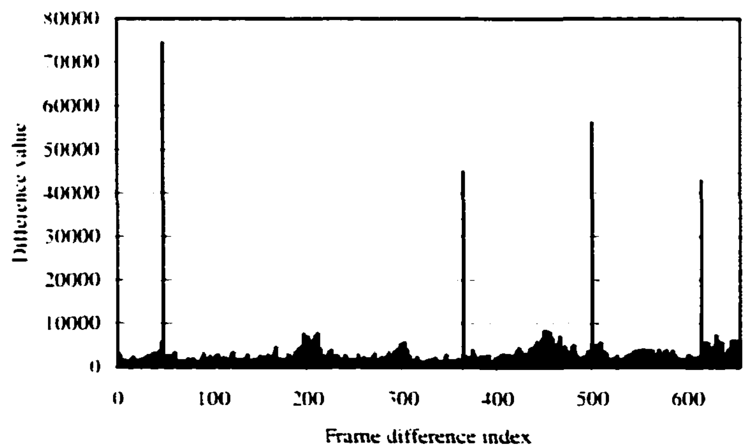

Fig. 14. The frame difference graph for the comedy video.

A close investigation to the algorithm's results for the class clip listed in TABLE 12 shows that a lot of needed KFs were not selected. This problem is the result of using a large threshold value with respect to the size of this MPEG clip. For instance, shot 0 has camera panning at its end that necessitates the use of two to three KFs to faithfully represent the shot; this need can be noticed by investigating the clip frames difference in Fig. 16. Reducing the value of the threshold is not a solution because this will affect the number of chosen key frames for large-dimension clips.

Fig. 15. Three key frames are ideally needed to abstract shot 1 in the comedy clip.

TABLE 12

KFs Selection for the class Clip Using δ_f =150.000 without any Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-442	443	1	0
1	443-804	362	1	443
2	805-1452	648	2	805, 1214
3	1453-1866	414	1	1453
4	1867-2306	440	1	1867
5	2307-2641	335	1	2307
6	2642-2792	151	1	2642

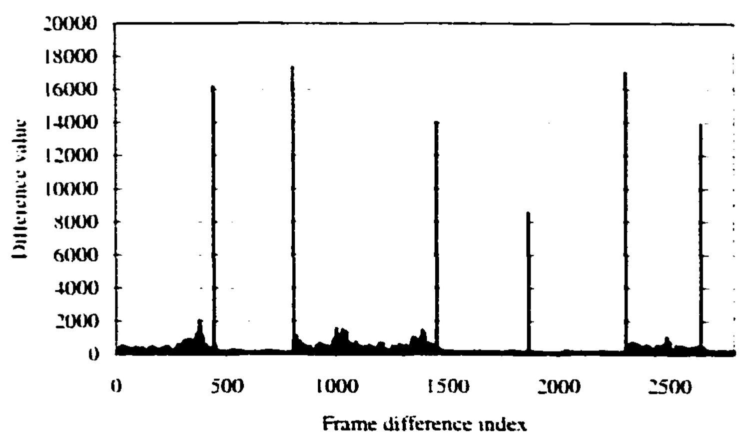

Fig. 16. The frame difference graph for the class video.

We can observe the main problem with the above-described algorithm through a study of its results especially for small-dimension clips (176x120 and less). The source of the problem is the use of a fixed threshold value for all the clip sizes. Our criterion to determine if a new frame can be added to the representative set is the accumulated frames differences and this value is function of the video dimension. The chosen threshold works well for large-dimension clips (320x240 and larger) but it performs badly for small ones. In a general-purpose video retrieval database, we cannot assume any control over the dimension of input video streams. To be general and flexible enough, any new stream should be accepted and the chosen threshold has to be adapted according to the input video dimension. This enhancement to the basic algorithm presented above is the topic of the next section.

3.2.2 Using the First Level of Threshold Adaptation

As was shown in the previous section, there is a shortcoming in the proposed algorithm that can be stated as follow. Using a single threshold for all input video files is not appropriate in all cases. The selection algorithm depends upon the accumulated differences between DC frames and the difference itself is function of the size of the DC frame. So different sizes need different thresholds in order to reliably select the most appropriate and minimal set of representative frames. This problem motivates us to propose an enhancement to the basic algorithm in which the threshold used to determine the selection frequency (how many KFs are selected) becomes a function of the input video frame size. The algorithm gets the size information from the segmentation module and uses it to adapt the chosen threshold so that a different threshold is used for different frame sizes.

The proposed enhanced algorithm chooses the new threshold using a linear function of the size of the input file. This enhancement was implemented and a similar set of experiments was performed to judge the effectiveness of that approach. As mentioned in Section 3.2.1, threshold choice is the first step and hence we list, in TABLE 13, the overall results in terms of the total number of selected KFs. Again a threshold value equals to 150,000 seems to be a good choice, so we use that value as an initial threshold value (δ_t). Then, the algorithm chooses an adapted threshold value, δ_{at}, for each clip in accordance with its dimension.

TABLE 13

Number of Selected KFs as a Function of Threshold Values with First-level Adaptation

Video name	δ_t = 50000		δ_t = 150000		δ_t = 250000		δ_t = 350000	
	# of KFs	Ave / shot	# of KFs	Ave / shot	# of KFs	Ave / shot	# of KFs	Ave / shot
soccer	31	10.3	13	4.3	8	2.7	7	2.3
racing-boats	29	4.8	14	2.3	8	1.3	8	1.3
action-movie	13	4.3	5	1.7	4	1.3	3	1.0
carton	40	4.4	18	2.0	14	1.6	11	1.2
celebration	29	14.5	11	5.5	7	3.5	5	2.5
comedy	29	5.8	11	2.2	8	1.6	6	1.2
ads	51	4.6	22	2.0	15	1.4	12	1.1
class	61	8.7	24	3.4	14	2.0	12	1.7
news-cast	148	6.4	63	2.7	42	1.8	32	1.4
conf-discussion	193	9.7	71	3.6	47	2.4	38	1.9
tv-show	353	4.8	142	1.9	106	1.5	92	1.3
Total Average		**7.14**		**2.88**		**1.91**		**1.54**

One important observation comes from comparing TABLE 13 with TABLE 7 and TABLE 8 is that the total averages of the number of selected key frames per shot (taken over all the 11 clips) are larger in TABLE 13 than their corresponding values in the other two tables. The above observation is a proof of the success of the proposed first-level adaptation algorithm in selecting more key frames (especially in the case of small-dimension clips) to better represent the video data. To illustrate the outcome of applying this modification, a set of tables similar to those given in the last section is presented in this section reporting the selection results for each individual clip. Results for some long clips are also given in APPENDIX A.

The size of the soccer clip is taken as a reference size in which all other adapted thresholds are calculated with respect to it. For TABLE 14 through TABLE 16, the algorithm detects the relatively small size of these clips and hence reduces the threshold value which leads to a general increase in the number of selected KFs (compared with corresponding numbers in the last section). This is true particularly for active shots such as shot 4 in TABLE 15.

TABLE 14

KFs Selection for the action-movie Clip Using δ_{al} =125.000 with First-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-51	52	2	0, 36
1	52-104	53	2	52, 86
2	105-177	73	1	105

TABLE 15

KFs Selection for the carton Clip Using δ_{al} =100.757 with First-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-48	49	1	0
1	49-83	35	2	49, 76
2	84-133	50	2	84, 97
3	134-151	18	2	134, 149
4	152-234	83	7	152, 156, 172, 182, 197, 209, 222
5	235-244	10	1	235
6	245-254	10	1	245
7	255-266	12	1	255
8	267-330	64	1	267

TABLE 16

KFs Selection for the comedy Clip Using δ_{al} =125.000 with First-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-48	49	1	0
1	49-364	316	4	49, 146, 206, 285
2	365-500	136	3	365, 441, 482
3	501-614	114	2	501, 570
4	615-654	40	1	615

KFs selection results for the class clip are given in TABLE 17 where the total number of selected KFs is 24 compared to only 8 in TABLE 12. This confirms the effectiveness of the proposed first-level adaptation algorithm in which it avoids skipping required KFs for small size clips like this one. More comments are worth noting. Shot 0 has some camera work that calls for more than one KF but the length of that shot is the main factor that causes the choice of five KFs. It is obvious from the frame difference diagram, Fig. 16, that there is some activity in shot 2 that requires more than one KF but again the length of that shot biases the algorithm to choose many key frames (nine) for such a medium activity shot. The same analysis applies to shots 3, 4, and 5. Shots 1 and 6 are completely still so the choice of one KF for each is the right one and the algorithm succeeds in making this choice although shot 1 is a relatively long one.

TABLE 17

KFs Selection for the class Clip Using δ_{a1} =29.166 with First-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-442	443	5	0, 171, 305, 364, 388
1	443-804	362	1	443
2	805-1452	648	9	805, 870, 990, 1028, 1093, 1204, 1325, 1384, 1439
3	1453-1866	414	2	1453, 1814
4	1867-2306	440	2	1867, 2280
5	2307-2641	335	4	2307, 2399, 2494, 2622
6	2642-2792	151	1	2642

In the last section, many of the necessary KFs have been skipped especially for small-dimension clips but this is not the case when applying the first-level adaptation algorithm. In a nutshell, the use of the first-level adaptation mechanism solves the problem of the input size but there are other problems that are still unsolved. The topic of the next section is to address the problems with the first-level adaptation algorithm and propose an effective and efficient strategy to overcome these shortcomings.

3.2.3 Using The Second Level of Threshold Adaptation

As discussed in the previous section, although the use of one-level adaptation solves the problem with various input file sizes the algorithm still has two main problems:

- It tends to choose more key frames in case of shots that have large number of frames even if they are inactive.
- It sometimes fails to select the required number of KFs in order to faithfully represent very active shots.

To tackle the above problems, we propose a second enhancement to the algorithm introduced in the last section by introducing a second level of adaptation. This second level is a shot-by-shot adaptation strategy that changes each shot threshold on the fly. That is to say, each shot in a certain video stream will get a threshold value in proportion to its activity level. So that active shots get lower threshold values in order to increase the selection pressure; thus, choosing more KFs to represent them. On the other side, low activity shots get higher threshold values to reduce the sampling rate and prevent selecting too many redundant key frames. We first measure the shot activity before invoking the selection algorithm then an adaptation heuristic is used to adapt each shot threshold based on the amount of activities found in that shot. As will be illustrated, the results of applying this technique are very good and it solves the problems with the use of only one level of adaptation.

Again threshold selection is the first step and some of the results are listed in TABLE 18. We select δ_t = 150,000 and the algorithm is allowed to adapt it to a value called δ_{a2}.

TABLE 18

Number of Selected KFs as a Function of Threshold Values with Second-level Adaptation

Video name	δ_t = 150000		δ_t = 250000	
	# of KFs	Ave / shot	# of KFs	Ave / shot
soccer	16	5.3	10	3.3
racing-boats	15	2.5	10	1.7
action-movie	6	2.0	4	1.3
carton	22	2.4	16	1.8
celebration	11	5.5	7	3.5
comedy	7	1.4	6	1.2
ads	19	1.7	14	1.3
class	14	2.0	10	1.4
news-cast	51	2.2	37	1.6
conf-discussion	49	2.5	33	1.7
tv-show	121	1.7	93	1.3
Total Average		**2.66**		**1.82**

To illustrate the effectiveness of the two-level adaptation mechanism, we start by discussing the differences between TABLE 18 and TABLE 13. For the first four clips and using δ_t=150,000, the two-level adaptation algorithm chooses more KFs (for each of these clips) than the corresponding numbers in TABLE 13. The algorithm detects the amount of activity in each shot and hence adjusts the selection frequency on a shot-by-shot basis. Each of the last five clips got total numbers of KFs in TABLE 13 that are larger than the corresponding values obtained in TABLE 18 that uses the two-level adaptation algorithm. Most of the last five clips are low activity ones and the decision of the algorithm to reduce the selection frequency is quite appropriate. The total average of TABLE 18 is 2.7 that is smaller than the total average we got in TABLE 13 (2.9) that uses only one level of adaptation. Thus, the two-level adaptation solves the two shortcomings of the first level while producing at the same time a near optimal set of representative frames. Consequently, the two-level adaptation algorithm balances the trade-off and manages to select of the minimal number of key frames capable of accurately representing the whole video stream.

To study the behavior of the algorithm for each clip, a set of tables similar to those given in the last two sections is presented here. For all the considered clips, the first-level threshold (δ_{a1}) will be the same as in the previous section and the second-level threshold (δ_{a2}) will be computed for each shot based on its level of activity. Moreover, in these tables we include another parameter that is the AI (Activity Index) of each shot. This parameter is equal to the summation of frame differences all over a specific shot divided by the number of frame differences in that shot. Flashlights are discarded as explained before. Results for some other longer clips are given in APPENDIX A.

The heuristic used in the second level adaptation process is to categorize a shot into one of three categories according to its activity index. These categories are low activity, medium activity, and high activity. Through a large number of experiments and observations of the video clips and their frame difference graphs, we define the value of two thresholds. Any shot with normalized activity index (normalized with respect to its size) less than the first threshold is categorized as a low activity one. A shot with a normalized activity index between the two thresholds is considered a medium activity shot. Otherwise, the shot is considered a high activity one. The difference between the activity levels of two shots of the carton clip is shown in Fig 17.

TABLE 19

KFs Selection for the action-movie Clip Using δ_{a1} =125,000 with Second-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs	AI	δ_{a2}
0	0-51	52	3	0, 31, 45	4621	93750
1	52-104	53	2	52, 86	3204	125000
2	105-177	73	1	105	1261	250000

The relative high activity of shot 0 in TABLE 19 causes a reduction in that shot threshold and an increase in the number of selected KFs compared to the same shot in TABLE 14. For the last shot, the algorithm increases its threshold upon the detection of its low activity nature but the number of selected KFs remains the same because it is only one frame (the minimum).

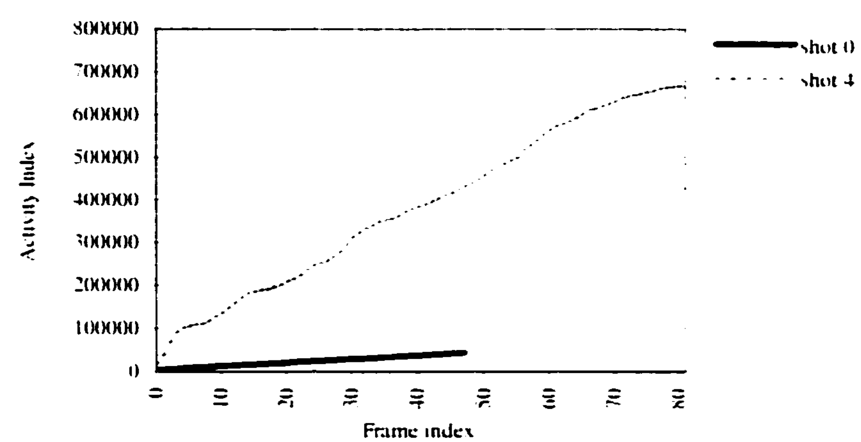

Fig. 17. The activity diagram for shots 0 and 4 of the carton clip.

TABLE 20

KFs Selection for the carton Clip Using δ_{a1} =100.757 with Second-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs	AI	δ_{a2}
0	0-48	49	1	0	916	201514
1	49-83	35	3	49, 73, 80	5826	75567
2	84-133	50	3	84, 91, 128	3523	75567
3	134-151	18	2	134, 147	6425	75567
4	152-234	83	9	152, 155, 165, 177, 184, 196, 205, 212, 223	8157	75567
5	235-244	10	1	235	3996	75567
6	245-254	10	1	245	681	201514
7	255-266	12	1	255	996	201514
8	267-330	64	1	267	1286	201514

In TABLE 20, the remarkable increase in the activity of shots 1, 2, and 4 causes the algorithm to increase the number of select KFs for each one of them compared to the same shots in TABLE 15 obtained by using the single level of adaptation. Other shots got the same number of KFs as their counterparts in TABLE 15. The difference in activity is remarkable from investigating the activity diagram in Fig. 17 where the curve representing shot 4 has a much higher monotonic rate than that of shot 0. The activity diagram shows this activity gap between the two shots; thus, supporting the decision made by the algorithm to increase the sampling frequency for higher activity shots.

The efficiency of the proposed shot-by-shot adaptation mechanism is evident in TABLE 21 in which the algorithm decreases the sampling frequency for low activities shots and hence reduces the number of KFs selected for them. For instance, the number of chosen KFs for shot 2 is reduced from 3 (see TABLE 16) to 2. The total KFs selected for this clip is 7 compared to 11 without using the second level of adaptation that gives us a total reduction of about 45%. This was achieved while keeping the sufficient number of KFs required to properly representing each shot.

TABLE 21

KFs Selection for the comedy Clip Using δ_{a1} =125.000 with Second-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs	AI	δ_{a2}
0	0-48	49	1	0	1418	250000
1	49-364	316	2	49, 206	1557	250000
2	365-500	136	2	365, 481	2077	250000
3	501-614	114	1	501	1730	250000
4	615-654	40	1	615	2966	125000

TABLE 22

KFs Selection for the class Clip Using δ_{a1} =29.166 with Second-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs	AI	δ_{a2}
0	0-442	443	3	0, 304, 386	297	58332
1	443-804	362	1	443	79	58332
2	805-1452	648	5	805, 990, 1092, 1323, 1435	367	58332
3	1453-1866	414	1	1453	80	58332
4	1867-2306	440	1	1867	72	58332
5	2307-2641	335	2	2307, 2493	279	58332
6	2642-2792	151	1	2642	102	58332

The reduction of the total number of selected KFs from 24 in TABLE 17 to 14 in TABLE 22 is an obvious proof of the success of the proposed shot-by-shot adaptation policy. Take shot 2 as an example, the algorithm selects only 5 KFs compared to 9 in

TABLE 17 which is about 45% reduction for that specific shot while the total reduction for the whole clip is about 42%. This reduction is mainly due to the adaptive nature of the algorithm in which it increases the selection frequency upon the detection of high activity and reduces it in cases of low activity shots such as most of the shots in this clip.

3.3 Using Luminance Differences of Successive DC Frames

In this section, another algorithm to select key frames from segmented video shots is introduced. The input video stream is segmented using the methodology described in CHAPTER II then the algorithm is applied to choose key frames that are able to represent each shot. The algorithm could be described using the following steps:

- *Initialize the representative set to be empty and select the first frame as the first element in that set.*
- *Initialize $i = 0$ and flashIndex to the last detected flash position plus one.*
- *For $j = 1$ To $N-1$ do {*

  if (j = (flashArray[flashIndex] + 1))

  flashIndex = flashIndex+1

  continue

  if (Diff(i , j) > δ_1)

  $i = j$

  Add *j* to the set of selected KFs

  Increment the number of selected KFs

  }

Where

$$Diff(i,j) = \left| \sum_{k=0}^{M-1} DC_k(i) - \sum_{k=0}^{M-1} DC_k(j) \right| \tag{19}$$

N: The number of frames in a shot.

flashArray[ ]: An array containing indexes to flashlight positions.

δ_1*:* An initial threshold that determines the frequency of sampling.

DC_i*:* The DC of the luminance channel at location i.

M: The number of DC terms in a DC frame.

This algorithm bears some similarity to the algorithm proposed in Section 3.2.3 but it uses direct frames differences instead of the accumulated frames differences, thus we call it ALD (Absolute Luminance Difference) algorithm. ALD has been implemented and based on the previous experience from implementing the AFS set of algorithms the same first level of adaptation was included. The second level of adaptation in ALD uses a different criterion to adapt the threshold on a shot-by-shot basis. In Section 3.2.3 we used the shot activity in the second level of adaptation but here we use a frame VI (Variance Index), the shot standard deviation, instead. The decision of using a frame variance comes from the nature of the selection method used. We use the direct difference between two consecutive frames and select another frame as a key frame if that difference is larger than a certain threshold. The problem here is that, we cannot use the same value of that threshold all over the whole video stream because different shots exhibit different lighting conditions and amount of activity. In an attempt to avoid the effect of these different conditions on the accuracy of the selection algorithm, we calculate the standard deviation of the shot (we call it the variance index) while discarding flashlight values as done before. The variance index is then used to categorize the shot into a number of categories: e.g., very high variance and very low variance shots. Then, the shot threshold is adapted according to the category each shot belongs. For instance, the shot threshold is reduced in case of low variance in order to select more KFs to properly represent the shot and to account for the fact that there are small differences between DC frames of that shot.

We start by performing similar experiments to those in the previous sections to determine the best values for the threshold. Then we use the best value obtained (15,000) and apply the two-level adaptation algorithm to the 11 clips used before. The overall results are shown in TABLE 23 as the total number of selected KFs for the best threshold value. Detailed results for each clip are given the same way we followed in the last section while the results for some other longer clips are listed in APPENDIX A.

TABLE 23

Number of Selected KFs with δ_t = 15.000 Using ALD Algorithm

Video name	δ_t = 150000	
	# of KFs	Ave / shot
soccer	7	2.3
racing-boats	18	3.0
action-movie	5	1.7
carton	16	1.8
celebration	9	4.5
comedy	5	1.0
ads	19	1.7
class	13	1.9
news-cast	66	2.9
conf-discussion	57	2.9
tv-show	97	1.3
Total Average		**2.3**

TABLE 24 and TABLE 25 report comparable results to their counterparts in Section 3.2.3. We can notice the general tendency of this algorithm towards selecting fewer KFs in which their distribution may not be uniform over the length of the shot as in shot 4 of TABLE 25.

TABLE 24

KFs Selection for the action-movie Clip with δ_{a1} =12.500 Using ALD Algorithm

Shot index	Range	# of frames	# of KFs	Index of selected KFs	VI	δ_{a2}
0	0-51	52	2	0, 50	539	10000
1	52-104	53	2	52, 85	829	10000
2	105-177	73	1	105	111	10000

TABLE 25

KFs Selection for the carton Clip with δ_{a1} =10.075 Using ALD Algorithm

Shot index	Range	# of frames	# of KFs	Index of selected KFs	VI	δ_{a2}
0	0-48	49	1	0	182	8060
1	49-83	35	1	49	583	8060
2	84-133	50	2	84, 129	794	8060
3	134-151	18	2	134, 151	1049	8060
4	152-234	83	5	152, 153, 156, 178, 183	1156	10075
5	235-244	10	1	235	1277	10075
6	245-254	10	1	245	338	8060
7	255-266	12	1	255	124	8060
8	267-330	64	2	267, 286	578	8060

All shots in TABLE 26 were categorized as slightly low variance shots and hence their respective shot thresholds were reduced in order to capture more KFs into representative sets. Nevertheless, the algorithm selects only one KF to represent each shot and these results render the performance of the algorithm bad in this clip. For instance, shot 1 required 3 KFs to be properly represented, see Fig. 15. The low variance nature (small variation around the average) of all the shots in the comedy clip can be depicted from investigating Fig. 18.

TABLE 26

KFs Selection for the comedy Clip with δ_{a1} =12.500 Using ALD Algorithm

Shot index	Range	# of frames	# of KFs	Index of selected KFs	VI	δ_{a2}
0	0-48	49	1	0	202	10000
1	49-364	316	1	49	156	10000
2	365-500	136	1	365	178	10000
3	501-614	114	1	501	279	10000
4	615-654	40	1	615	307	10000

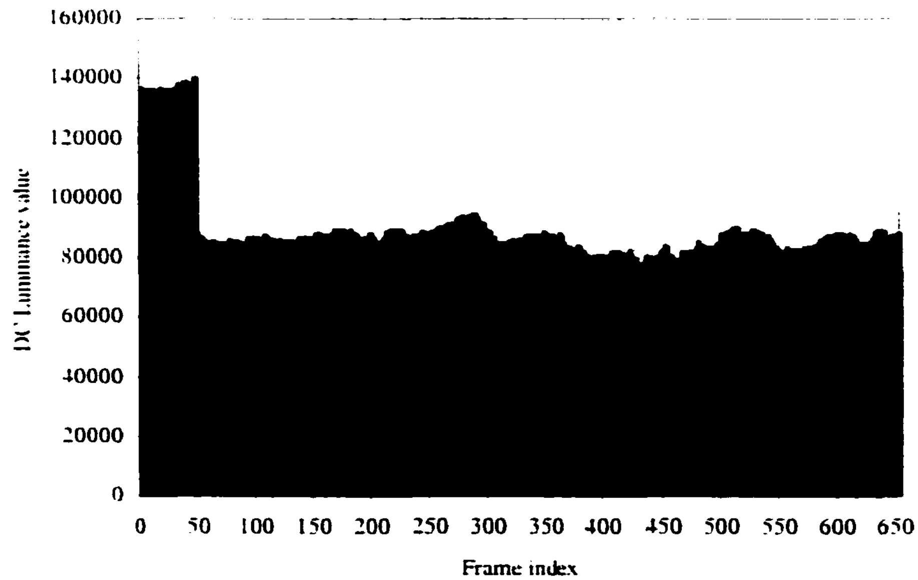

Fig. 18. The value of the DC luminance frames in the comedy clip.

In TABLE 27, the algorithm chooses three KFs to represent shot 0 which is a good choice as can be observed from Fig. 19. Moreover, the results in TABLE 27 are almost similar to those in TABLE 22 except that the number of KFs selected for shot 2 is just 4 instead of 5. The decision to select only 4 KFs to represent shot 2 is an appropriate one because of the intermediate activity of that shot but the distribution of the selected KFs is not uniform all over the length of the shot.

TABLE 27

KFs Selection for the class Clip with δ_{a1} =2.916 Using ALD Algorithm

Shot index	Range	# of frames	# of KFs	Index of selected KFs	VI	δ_{a2}
0	0-442	443	3	0, 362, 377	513	2332
1	443-804	362	1	443	32	1458
2	805-1452	648	4	805, 810, 1369, 1420	176	2332
3	1453-1866	414	1	1453	24	1458
4	1867-2306	440	1	1867	19	1458
5	2307-2641	335	2	2307, 2484	84	1749
6	2642-2792	151	1	2642	57	1458

Fig. 19. The three KFs selected by ALD to represent shot 0 of the class video.

3.4 Performance Comparison

We have proposed two sets of algorithms in this chapter, the AFS and the ALD algorithms, and in this section we will attempt to give the pros and cons of each one focusing on the last algorithm of each set that uses two-level adaptation.

- The AFS is generally biased towards selecting more KFs as the length of the shot increases. We introduce the second level of adaptation to effectively alleviate this natural bias. On the other hand this bias is not exhibited by the ALD.
- In general, ALD has a tendency to select less key frames per shot compared to those selected by the AFS. This is obvious from investigating TABLE 18 and TABLE 23. Fig. 20 shows the number of selected KFs for each shot of the carton clip using both AFS and ALD.

- The ALD is more sensitive to lighting changes than the AFS. An investigation of the results for the ads clip in APPENDIX A supports that conclusion.
- The distribution of KFs selected by the ALD may not be uniform over the length of the shot.
- We can conclude that the performance of the AFS, measured in terms of the number of selected KFs, is slightly better than that of the ALD as it captures the notion of activity in each shot in a better way. Therefore, the AFS takes more appropriate decisions in selecting each shot threshold and hence in determining the frequency of sampling for each individual shot. Those decisions have a direct impact on its effectiveness and applicability.

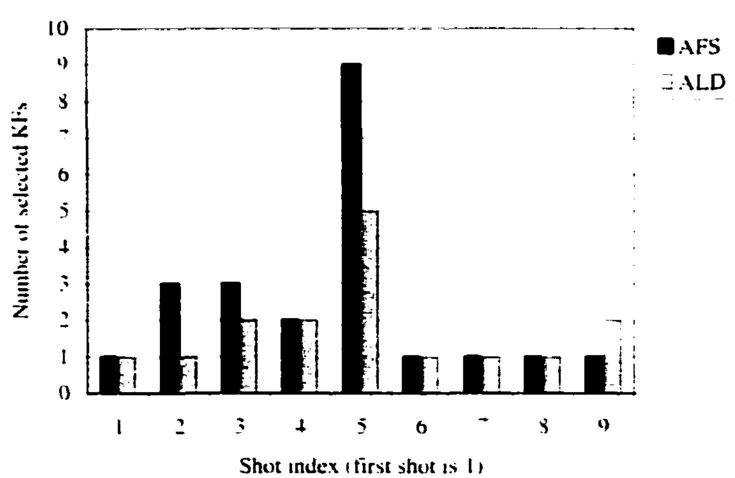

Fig. 20. The number of selected KFs for each shot of the carton clip using AFS and ALD.

3.5 Conclusion

In this chapter, we have introduced two algorithms to effectively and efficiently select KFs (Key Frames) form segmented videos, an essential task in fully content analysis systems. At first, we present the basic form of the AFS algorithm followed by two enhancements that overcome its original shortcomings. The first one adjusts the value of the clip threshold based on its dimension. The second enhancement measures the amount of activity in each shot then adapts the threshold for each shot on the fly based on its activity level. The application of these enhancements renders us a two-level adaptation algorithm that can accurately perform the task of KFs selection and improve the results of the basic algorithm by about 40% in most clips. Moreover, the two-level adaptation algorithm produces a near optimal set of representative frames, compared to the ground truth, that can be used to summarize the video data or as access points.

The second algorithm, ALD, applies a similar first-level adaptation strategy to the one used by the AFS. The second-level adaptation policy in that algorithm uses a statistical criterion to adapt the threshold on a shot-by-shot basis. Performance comparisons of both algorithms were given and we conclude the superiority of the AFS over the ALD algorithm. From our experimentation, the proposed algorithms proved insensitive to lighting changes, an evidence of robustness. Moreover, managing to select a near optimal set of KFs from a clip with 6139 frames in less than 2 seconds while running on a Sparc ultra 60 machine is an evidence of the efficiency of the proposed algorithms.

It is worth noting at this point that the dependency upon the format of the input compressed video (MPEG videos in our case) is isolated into the shot detection module. This means that the proposed KFs selection module can work with any type of compressed video formats provided that the same information is passed to it from the shot detection stage.

Using the techniques developed in CHAPTER II and CHAPTER III, we managed to design and implement an effective and efficient system for analyzing MPEG compressed video [25]. The topic of the next chapter is how to use the produced summary to effectively construct indexes for video data.

CHAPTER IV

DERIVING METADATA CONTENT INDEXES

In the previous two chapters, we devised techniques to accomplish the task of analyzing digital video data. At first, the input video data are segmented then key frames are selected to represent each shot using the proposed analysis techniques. In addition, the concept of content-based access to video data has been explained. This access method requires effective indexing schemes to facilitate the retrieval of the intended data. This chapter proposes two techniques [28] to be used in generating color and texture indexes for digital video data. The first one of them derives color indexes for the selected key frames directly from compressed data while the second scheme is used to produce texture indexes to represent the selected key frames. The feature vectors (color and texture) are then stored as metadata for this particular input clip. Experimental results show the effectiveness of the proposed indexing schemes and evaluate their efficiency.

4.1 Introduction

To facilitate access to large video databases, the stored data need to be organized and a straightforward way to do such organization is the use of index structures. In case of video databases we even need multi-dimension index structures to account for the multiple features used in indexing. Moreover, we are in need for tools to automatically or semi-automatically extract these indices for proper annotation of video content. Bearing in mind that each type of videos has its own characteristics, we also need to use multiple descriptive criteria in order to capture all of these characteristics.

As mentioned before, video data are rich sources of information and in order to model these data, the information content of the data has to be analyzed. Hence, the first step in indexing video databases (to facilitate efficient access) is to analyze the incoming video stream. Following the shot boundary detection and key frames selection tasks, the next step is to derive descriptive indexes from selected key frames in order to represent them then use the indexes as metadata. Any further similarity matching operations will be performed over these indexes and not over the original key frames data. In this chapter, two effective schemes for generating indexes to abstract digital video data are introduced. Both schemes works upon the key frames produced by the algorithms discussed before in CHAPTER III in order to generate low-level indexes to represent them. First, color histograms are derived and used as feature vectors to represent the color of the selected key frames. Then an effective methodology using the Gabor wavelet transform is employed to generate texture indexes. The two sets of feature vectors (color feature vectors and texture feature vectors) are then stored as metadata that is associated with the original video clip stored into the database.

The remainder of this chapter is organized as follows. We briefly review, in Section 4.2, a number of related approaches for indexing video data. The histogram technique used to derive color indexes is then illustrated in Section 4.3. Section 4.4 explains the concept of using the Gabor wavelet transform to extract texture indexes from selected key frames. Experimental results are given in Section 4.5 followed by conclusion in Section 4.6.

4.2 Related Work

Ideally, CBR (Content-Based Retrieval) of video data should be accomplished based on automatic extraction of content semantics but most of the current techniques only check the presence of semantic primitives such as objects or represent the content using low-level features. There are mainly two major trends in the research community to extract indices for proper video indexing and annotation. The first one tries to propose methods to automatically extract these indices while the second trend performs iconic annotation of video by manually (with human help) associating icons to parts of the video stream. One example of the latter trend can be found in [20]. This research uses a multi-layered representation to perform the annotation task where each layer represents a different view of video content. In this way, the indices support access at different levels of details based on the requirements of various applications. The system proposed in [20] has a directory workshop where iconic descriptors are created. The user can accumulate one or more icons in different palette. Icons can be dragged and dropped on a media time line to annotate the temporal media properties. Other techniques for CBR of video data have been reviewed in [9] and some of them use SQL-like query languages to formulate queries.

On the other hand, works on the first trend, automatic extraction of content indices, can be divided into three categories.

- The first one derives indices for visual elements (color, texture, shape, etc.) of a video frame by using image-indexing techniques.
- The second category extracts indices for camera motion (panning, zooming, etc.).
- The third category attempts to derive indices for region/object motion.

In the following, we will try to give an overview of various systems that supports automatic extraction of content indexes, where some of them may be categorized under only one of the above categories or may belong to more than one.

One technique to deduce indices for camera motion that works in uncompressed domain is introduced in [98]. At first, optical flow (motion vectors field) is calculated using block-matching technique and used to detect camera movements. Panning and tilting are detected by thresholding differences in absolute values of each motion vectors (Θ_k) and the modal vector (Θ) as described in equation (20). Zooming is detected by observing that vertical components of motion vectors in the first and last row of the frame have opposite signs so their difference exceeds the magnitude of both components. In the same paper [98], the authors suggest the possible use of motion vectors derived directly from compressed MPEG video data to detect camera operations.

$$\sum_{k=1}^{N} |\Theta_k - \Theta| \leq \Theta_p \tag{20}$$

Another research effort that can be categorized under the third category is introduced in [100]. This system analyzes key frames in order to detect major objects/regions within the frames and calls these objects key-objects (regions with coherent motion). The authors proclaimed that the process of segmenting semantic objects is very hard thus they rely on regions of coherent motion to define objects. Key-objects are detected using optical flow techniques where local motion estimation is performed first then a motion model is used afterwards. The authors show that the concept of key-objects can be used to select key frames and defines the similarity measure as the similarity among the visual attributes of these key-objects.

A new video model has been proposed in [42]. This model is based on the stratification concept that focuses on segmenting the video's contextual information into multiple layers. Another effort, an extension to the Four Eyes system [55], [56], [63] has been introduced in [86] to enable video browsers to learn by examples. This research is focused on video sequences of situation comedy and it uses high-level information (show's script and closed caption) to accomplish its task. A prototype system is presented in [75] that consists of three major stages, parsing, indexing, and browsing and retrieval. Parsing performs cut and camera works detection. Indexing is performed through a knowledge-based structure (frame-based database). Finally, the retrieval stage uses low-level features as indices while browsing employs Micons and hierarchical video magnifier techniques.

One more system for video CBR that belongs to the third category is introduced in [102] where an object segmentation algorithm is first applied to segment objects that are found in video frames. Color, edge, and optical flow are the three features used. Adjacent regions of salient objects are then grouped based on similarity in motion while the sizes and durations are used to eliminate noisy and unimportant regions. The authors proposed a framework for spatio-temporal video search by extending the concept of 2D strings to be used for video data.

A prototype system has been introduced in [22]. This system performs video segmentation on compressed data where cuts are detected in P and B frames by setting a threshold on the number of motion vectors in these frames. The features extraction stage employs color histograms, Gabor texture feature set, and motion histograms that are analogous to color histograms. A motion-based indexing technique is proposed in [69] where object trajectories are used to aid indexing. The system first detects objects, determines their motion patterns, then applies the wavelet transform to the motion coordinates and stores the coefficients as indices. It applies a motion segmentation algorithm to MPEG compressed video in order to perform the trajectories extraction step. The system is queried by sketching the required trajectory that is in turns analyzed and compared to the indices stored into the database using the R^{16} Euclidean norm.

The QBIC system developed at the IBM Almaden research center [31], [59] performs both image and video indexing and retrieval and it belongs to the first category. The system in general consists of two phases. the database population phase and the database query phase. In addition. the system uses more than one indexing criteria to facilitate different types of queries.

Another system called ViBE [14] is a browseable and searchable paradigm for organizing video data containing large number of sequences. The system first segments video sequences into shots then each shot is represented by a hierarchical structure known as the shot tree. The shots are then classified into pseudo-semantic classes that describe the shot content. Finally. the results are presented to the user in an active browsing environment using a similarity pyramid data structure. The similarity pyramid allows the user to view the video database at various levels of detail. The use of pseudo-semantic labels in that system provides a novel way of describing shots that can be used in browsing environments.

The Berkeley digital library project [8] has a large part of it focused on image and video analysis for retrieval and browsing. The aim is to improve the performance of the query stage by deriving feature vectors that have the following properties: rich descriptors. invariant under irrelevant variation. and associated with intuitive metrics.

A digital video library called Informedia [17]. [18]. [87] was developed at CMU. This system investigates the utility of speech recognition. image processing. and natural language processing techniques to improve indexing and searching of video libraries. The system uses image processing techniques working in compressed domain to reduce processing time. These imaging techniques are used also to achieve video segmentation. key frames extraction. and identifying and indexing features to support video data similarity matching.

4.3 Color Indexing

Color is a feature that is largely independent of view or resolution: thus. can be used effectively as an indexing methodology. One of the effective techniques that enable the use of this property in indexing is the color histogram. Given a discrete color space. the color histogram counts how many of each color occurs in the image. Histograms are also invariant to translation. rotation about a normal axis. and change slowly with rotation about other axes. change of viewing distance. and occlusion [78].

As mentioned before, the first step in any system performing CBR of video data is to segment the video stream into its constituent shots as described in CHAPTER II. For each shot, we try to generate an abstract form by selecting one or more representative frame(s). This selection has mainly two reasons.

- Reduce the amount of data to be indexed.
- Provide a framework for dealing with video content.

The next step is to derive content indexes from the selected key frames. In this section we describe the technique we employed to generate color feature vectors to represent key frames and hence the video shots.

The proposed technique for color feature extraction works on an abstract form that represents the original video data, the DC sequence. As pointed out before, that sequence is extracted directly from MPEG compressed video streams without the need for full decoding. Working on the DC sequence has the obvious advantage of considerably speeding up the indexing process by reducing the amount of data processed by the indexing stage.

The color space used by MPEG compression standard is the YCbCr where Y represents the luminance component and Cb and Cr represent the chrominance components. This color space is used basically to exploit a desirable characteristic of the human visual system in which the eye is more sensitive to the luminance component of the color than to its chrominance components [57]. That is why the luminance component is represented using a higher precession, in each Microblock (6 blocks) four blocks are Y, one block is Cb, and the last one is Cr. The extracted DC sequence is actually three sequences one for each component of the YCbCr color space. These three components of each frame need to be converted to the RGB color space in order to derive color indexes from them.

The color indexing module is given the three DC sequences stored into disk files in addition to the positions of the selected key frames for each shot. It then starts to randomly access these files to read only the color components of the selected key frames and convert the YCbCr values to the RGB color space. Each chrominance component is shifted by a constant value (128 is subtracted from chrominance values) then equations (21), (22), and (23) are used to generate the corresponding RGB values.

$$R = Y + Cr * 1.402 \quad (21)$$

$$G = Y - Cr * 0.71414 - Cb * 0.34414 \quad (22)$$

$$B = Y + Cb * 1.772 \quad (23)$$

After applying the color conversion formulas each of the produced values is clamped to make sure its range is between 0 and 255 (the 8-bit representation of the RGB color model). As we stated before, the MPEG compression standard uses different sizes to represent its color components. The above formulas require equal-dimension matrices for all of the Y, Cb, and Cr components. To enable the application of the conversion equations one value of each of Cb and Cr blocks are used in the conversion of their surrounding four Y blocks. Fig. 21 illustrates this association.

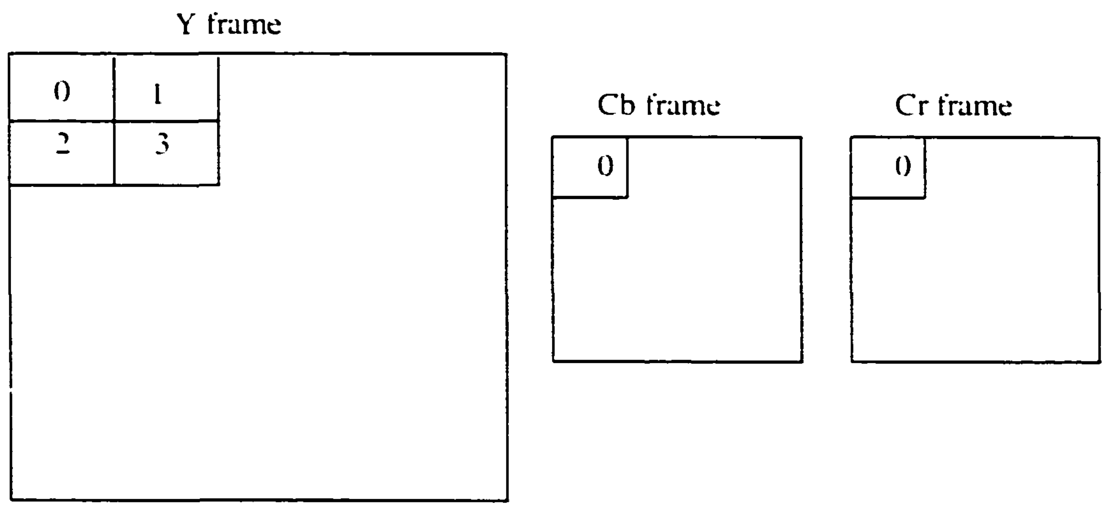

Fig. 21. The use of only one block of Cb/Cr component in conjunction with its surrounding four Y blocks in the conversion to the RGB color space.

Now, we have three 8-bit values that represent each DC block in the selected key frames (we are working on the DC frames and not the original decoded frames as mentioned before). Combining the values of the three RGB components and use them to construct a color histogram will produce a very huge histogram (2^{24}-bin). This huge histogram requires extensive storage spaces and computation resources that are not even practical using today's technology. To solve these problems while trying to achieve an accurate representation we need to use a lower dimension version of such histogram. A very important property of color histograms cited in the influential paper [78] is that histograms are insensitive to varying the representation precession. We based our reduction of the histogram dimension on this proven fact. This dimensionality reduction is achieved by selecting the most significant two bits from each color components forming a 6-bit code. This 6-bit code produces a 64-bin color histogram that is a good compromise between the use of the available computational resources and the representation accuracy. Fig. 22 shows an example of RGB values and the produced codeword.

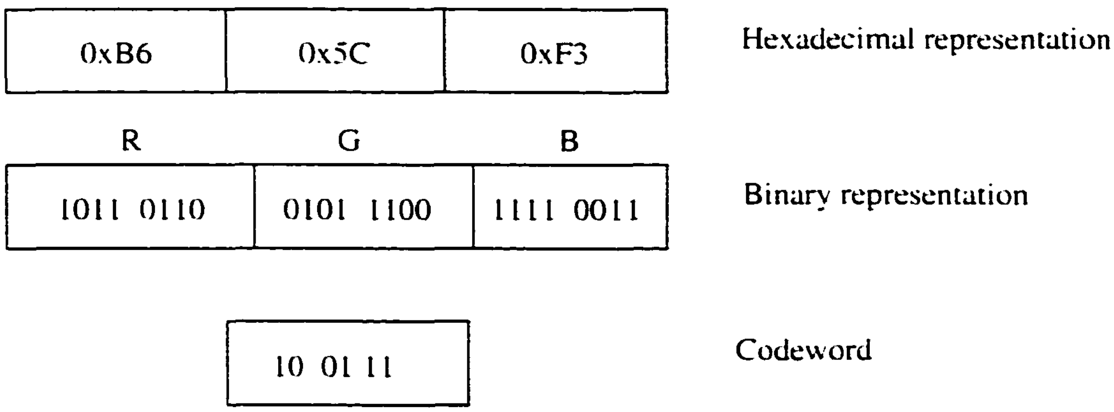

Fig. 22. Example of generating the codeword used to construct the color histogram from the RGB color values.

Each block is now represented by its codeword value (ranging from 0 to 63). A 64-bin color histogram is then constructed by counting how many of each codeword values occur in the frame. The calculated histogram for each key frame is stored in an array of integer values. We designed an object data structure to store all the metadata associated with evry shot. The information contained on that object is as follows:

- The number of key frames selected to represent this shot.
- The position of the selected key frames (represented as array of integers with the value of the ith element equals to the position of the ith key frame).
- The color histograms of all of the selected key frames are represented as 2D integer array. This array has the number of rows equals to the number of selected

key frames for that shot and the number of columns equals to the dimension of the color feature vector (64 in our case).

- The texture feature vectors for all the selected key frames are represented as 2D floating point array. This array has the number of rows equals to the number of selected key frames for that shot and the number of columns equals to the length of the texture feature vector (that is a function of the number of scales and orientation of the Gabor filter, see Section 4.4).

This object design is very flexible in which it allows us to extend the number of the representation features in a seamless way if we need to add more low-level description to the selected key frames to expand the space of supported query types.

The last issue we need to expound here is how to measure the similarity between two color histograms. One of the most effective methods to perform this task is the use of the normalized histogram intersection methodology and thus we adapt this method in determining histogram similarity. The similarity between two images (key frames) can now be represented by a scalar in the range from 0 to 1. With 0 indicates completely dissimilar and 1 indicated exactly similar. Equation (24) is used to calculate the normalized histogram intersection while Fig. 23 illustrates a very simple example of histogram intersection calculation.

$$H(I,M) = \frac{\sum_{i=1}^{n} \min(I_i, M_i)}{\sum_{i=1}^{n} M_i} \qquad (24)$$

Where

H(I,M): The similarity between an image *I* and a model image *M*.

n: The number of bins in the histogram (64 in our case).

I_i: The value of image I's histogram at ith position.

M_i: The value of image M's histogram at ith position.

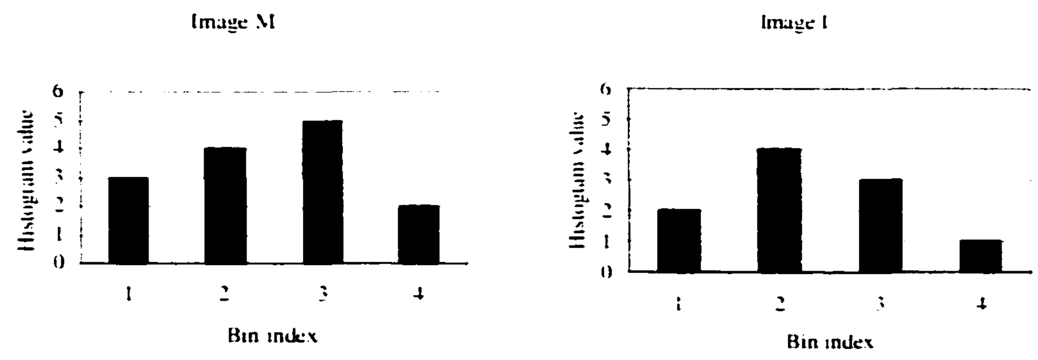

Fig. 23. Two simple histograms with the intersection value equals to 0.714.

4.4 Texture Indexing

Texture is an important feature of a visible surface where repetition or quasi-repetition of a fundamental pattern occurs. Texture features include coarseness, contrast, directionality, roughness, regularity, frequency and density [9]. It is a powerful discriminating feature presents almost everywhere in nature and it can be described according to their spatial, frequency or perceptual properties. Periodicity, coarseness, preferred direction, and degree of complexity, are some of the most perceptually salient attributes of texture. Feature spaces based on these attributes are particularity interesting for retrieval of multimedia data using texture similarity.

Different researchers proposed various statistical and structural properties to represent the texture feature of images. These techniques can be divided into three categories [9]:

- Space-base models (examples: auto-regression and stochastic models).
- Frequency-based models (examples: power spectrum and wavelet transform).
- Signature-based models (examples: textural energy, contrast, coarseness, and directionality).

It has been proven that the Gabor wavelet texture features perform better than other feature extraction methodologies [53]; thus, we decided to adapt this technique in generating texture indexes for the selected key frames.

The wavelet analysis is a scale-independent mean of analyzing signals [24]. It uses short windows at high frequencies and long windows at low frequencies. The notion of scale is introduced as an alternative to frequency, which leads to a time-scale representation. In Fourier transform, a signal is mapped into a time-frequency plane, whereas in wavelet transform, it is mapped into a time-scale plane. In wavelet analysis, a basic function called basic wavelet or mother wavelet is used as a window or a scale. At the same time, all possible scaling of the basic wavelet are used as basis functions derived from it. These basis functions, called wavelets, are generated by translating (shifting) and dilating (scaling) the basic wavelet in time.

Assuming that $g(x,y)$ is a mother Gabor wavelet function, its wavelets, $g_{mn}(x,y)$, can be calculated using equations (25) through equation (29).

$$g_{mn}(x,y) = a^{-m} G(x',y') \tag{25}$$

$$G(u,v) = \exp\left[-\frac{1}{2}\left(\frac{(u-W)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right)\right] \tag{26}$$

$$x' = a^{-m}(x\cos\theta + y\sin\theta) \tag{27}$$

$$y' = a^{-m}(-x\sin\theta + y\cos\theta) \tag{28}$$

$$\theta = n\pi / K \tag{29}$$

Where:

K: The number of orientations of the Gabor filter.

S: The number of scales of the Gabor filter.

a: Scale factor.

n: An integer ranges from 0 to K-1.

m: An integer ranges from 0 to S-1.

The design of the Gabor filter can be applied using equations (30), (31), and (32) as described in [53].

$$a = (U_h / U_l)^{\frac{1}{s-1}} \tag{30}$$

$$\sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln 2}} \tag{31}$$

$$\sigma_v = \tan\left(\frac{\pi}{2K}\right)\left[U_h - 2\ln\left(\frac{\sigma_u^2}{U_h}\right)\right]\left[2\ln 2 - \frac{(2\ln 2)^2\sigma_u^2}{U_h^2}\right]^{-0.5} \tag{32}$$

Where:

U_l: Lower frequency of interest.

U_h: Upper frequency of interest.

$W = U_h$.

The Gabor wavelet transform, $W_{mn}(x,y)$, can then be calculated by taking the 2D inverse Fourier transform of the multiplication of the image and the generated Gabor filter. The means (μ_{mn}) and the standard deviations (σ_{mn}) (given in equations (33) and (34) respectively) of the magnitude of the transform are used as elements of the texture feature vectors (F) shown in equation (35).

$$\mu_{mn} = \iint \left|W_{mn}(x,y)\right| dx\,dy \quad (33)$$

$$\sigma_{mn} = \sqrt{\iint \left(\left|W_{mn}(x,y)\right| - \mu_{mn}\right)^2 dx\,dy} \quad (34)$$

$$F = \left[\mu_{00}\sigma_{00}\mu_{01}\ldots\ldots\ldots\ldots\ldots\ldots\mu_{35}\sigma_{35}\right] \quad (35)$$

After deriving the feature vector we need to define a method of measuring the similarity between feature vectors. One suggested criteria to be used in the similarity matching stage of our content-based retrieval system is to calculate the summation of all the differences between corresponding mean and standard deviation values normalized to the standard deviation of the whole database. Thus, the similarity between feature vector i and feature vector j can be calculated using equations (36) and (37).

$$similarity(i,j) = \sum_m \sum_n d_{mn}(i,j) \quad (36)$$

$$d_{mn}(i,j) = \left|\frac{\mu_{mn}^{(i)} - \mu_{mn}^{(j)}}{\alpha(\mu_{mn})}\right| + \left|\frac{\sigma_{mn}^{(i)} - \sigma_{mn}^{(j)}}{\alpha(\sigma_{mn})}\right| \quad (37)$$

Where:

$\alpha(\mu_{mn})$: The standard deviation of the mean features over the entire database.

$\alpha(\sigma_{mn})$: The standard deviation of the σ features over the entire database.

For each selected key frame we derive a feature vector (F) to represent this frame and store these features vectors into the fourth element (the 2D array of floating point numbers) of the shot information object described in Section 4.3. This object has general information about the shot (the number of key frames, their positions, etc.) in addition to the color feature vectors and the texture feature vectors for all the selected key frames.

Each shot will be represented by an instance of that object and the metadata for a complete video clip is stored into a disk file as an array of such objects.

4.5 Experimental Results

In this section, some of the experimental results obtained from the video segmentation and the key frames selection stages will be used in testing the performance of the employed indexing scheme. In the following, we will focus on a reduced set of the clips segmented in TABLE 3. This reduced set contains the action movie clip and the carton clip in which their key frames selection results using the AFS algorithm were given in TABLE 19 and TABLE 20 respectively.

Let us consider the movie clip first. The key frames selection algorithm chooses three key frames to represent the first shot, two for the second, and only one frame for the last shot. The selected key frames for the first shot is shown in Fig. 12.

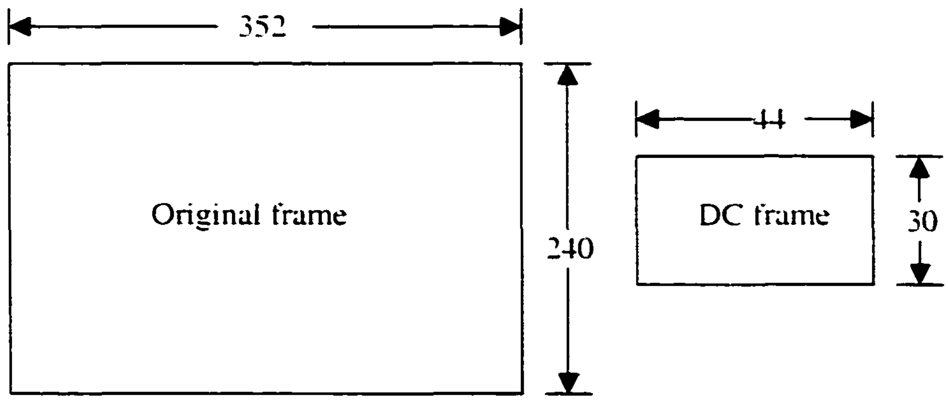

Fig. 24. The original frame size is to the left while the DC frame is on the right.

The DC sequence of the whole clip were extracted by the parsing module and stored into disk files (one for each color component of YCbCr). Once the indexing module is invoked, it starts to randomly access these files to retrieve the values of the selected key frames only. The size of the movie clip frames is 352x240 while the size of the DC frames is 44x30. Fig. 24 shows schematic representations of the two types of frames. The reduction of frame size is a key to the efficiency of the indexing and retrieval systems.

The color indexing module then converts the read YCbCr values to the RGB color space so we end up with a DC frame representation in the RGB color model (see equations 21-23). Each term of the converted frames is represented by an integer number with its three least significant bytes encode the values of R, G, and B color components respectively. To generate the color histogram out of these frames representation, a 6-bit code is formed from the 2 most significant bits of each color component. Then, according to the frequency of occurrence of each possible value in the derived code words, a 64-bin histogram is constructed (see Section 4.3) for each of the key frames chosen. The color histogram obtained from the first KF (the one with zero index) of the first shot of the movie clip is shown in Fig. 25.

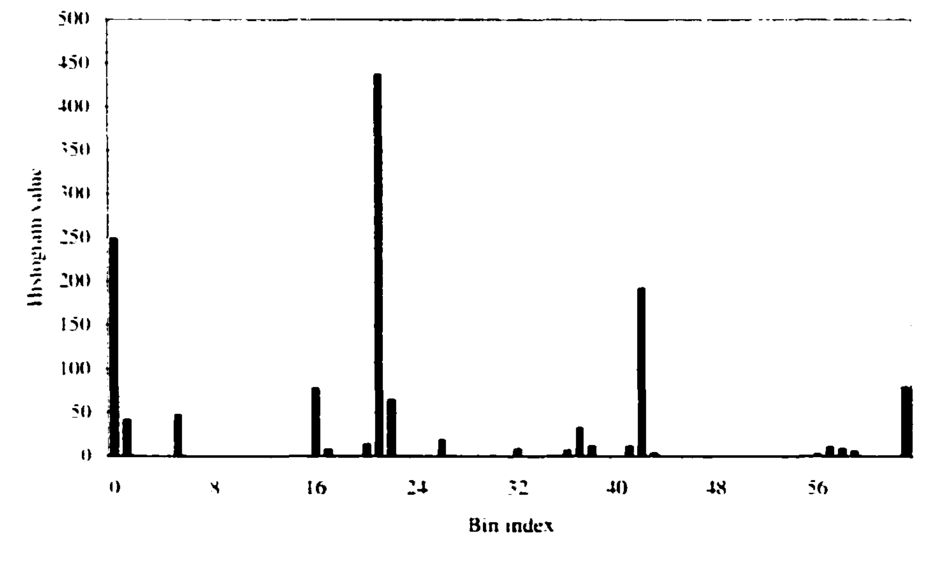

Fig. 25. The histogram of the first selected key frame of the first shot of the movie clip.

As mentioned in Section 4.3, the data structure used (the object representation of the shot) employs a 2D array of integer to represent each histogram with column length equals to the number of bins in the histogram (64 in our case). For a set of key frames selected to abstract a shot each row in that array is used to hold the histogram of an element of that set. Thus, the number of rows in this 2D array is equal to the number of chosen key frames. To test the results of the employed color similarity matching technique, we calculate the similarity among all the selected key frames of the movie clip using equation (24) and list the results in TABLE 28. It is clear from investigating the results in TABLE 28 that the similarity increases for frames belong to the same shot and decreases (as anticipated) for frames across different shots. The similarity values among the first shot key frames and the key frame of the last shot are higher than their values with the key frames of the second shot because both the first and the last shot are depicting the same character. Note also the symmetry of the similarity matrix in TABLE 28 and that all the diagonal values are 1.

TABLE 28

Color Similarity Values among all Selected Key Frames for the movie Clip

	KF0	KF1	KF2	KF3	KF4	KF5
KF0	1.0	0.8984848	0.9159091	0.7060606	0.67651516	0.8469697
KF1	0.8984848	1.0	0.91742426	0.7181818	0.6969697	0.8939394
KF2	0.9159091	0.91742426	1.0	0.6840909	0.6515151	0.87954545
KF3	0.7060606	0.7181818	0.6840909	1.0	0.9	0.6840909
KF4	0.67651516	0.6969697	0.6515151	0.9	1.0	0.6757576
KF5	0.8469697	0.8939394	0.87954545	0.6840909	0.6757576	1.0

With respect to extracting the texture using the Gabor wavelet technique there is a subtle issue we need to explain. We can clarify this issue be restating the definition of the term texture which is the repetition or quasi-repetition of patterns within an image. This means that to effectively capture the texture of the selected key frames we need to devise an indexing technique that represents the texture regardless of the frequencies of its patterns. If we use the DC frames (used in all other processing stages) we will only consider the DC term of each block and discard all AC coefficients of the DCT transform. Considering only the average intensity of the block and ignoring all AC coefficients will lead to poor representation of texture pattern specifically high frequency ones. For that reason we decided to derive the texture indexes from the original decoded frames not from the DC frames. We justify this decision that requires complete decoding of the selected key frames by the need to accurately represent the texture feature in order to achieve effective indexing scheme.

At first, the texture indexing module invokes a decoding module to fully decode the selected key frames and store the decoded frames into a separate disk file (one for each key frame). Subsequently, the texture indexing calls the Gabor wavelet module to calculate the texture feature vectors. The texture feature vector length is 48 floating-point elements (for $S = 4$ and $K = 6$) and it is stored in a one-dimensional array. The whole set of feature vectors representing the shot is stored into a two dimensional array that has a number of rows equals to the number of selected key frames. For instance, the feature vector of the first KF of the first shot of the movie clip is stored into 48-element array with the values of smallest and largest elements of 38566.17 and 2890778 respectively.

Texture calculation is a very time consuming process, and for that reason we tried to profile the execution time spent in each module to determine the part that consumes the bulk of the execution time. Fig. 26 shows the cumulative execution time of each module (the time spent by the module including the time spent in calling any function within it). We can observe that the module that calculates the Gabor transform (including the calculation of FFT (Fast Fourier Transform) and its inverse is the most time consuming part of the texture calculation. As Fig. 26 also indicates, the calculation of the texture vector can take more than 20 seconds for a single key frame that is a considerable amount of time and hence we try to study the effect of changing the Gabor transform parameters (mainly the scale and orientation) on the total execution time. TABLE 29 shows the total execution time of the texture stage as a function of varying scales and orientations.

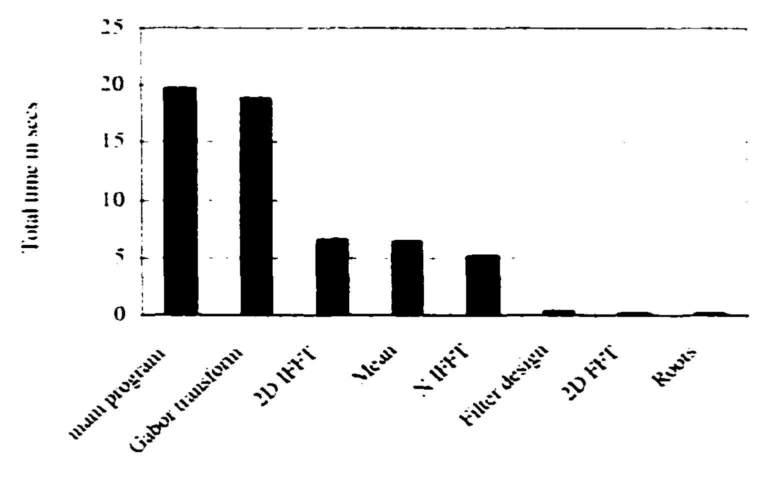

Fig. 26. Execution times of various modules of the texture extraction stage applied to the movie clip (obtained using a Matlab version of the code).

TABLE 29

The Execution Time of the Texture Extraction as a Function of Scales and Orientations of the Gabor Wavelet Transform

S	3	3	3	4	4	4	5	5	5
K	4	5	6	4	5	6	4	5	6
Time (sec)	21.1	26	31	27.6	34.4	41.3	34.3	42.9	51.3

We choose $S = 4$ and $K = 6$ as a compromise between accuracy and computational time required. Other combinations of values of S and K can be considered based on their effect on the retrieval performance.

The experiments performed above on the movie clip will be applied to the carton clip and the results are discussed below. The results of applying the key frames selection algorithm upon the carton clip were given in TABLE 20. Fig. 27 shows the nine key frames selected to represent the fifth shot of the carton clip.

Fig. 27. The nine selected key frames for the fifth shot of the carton clip.

The color indexing scheme is then applied and an example of the generated color histogram is given in Fig. 28. The color similarities among the 7 key frames selected for the first three shots of the carton clip are also shown in TABLE 30. The texture extraction results for the same clip are reported in the following paragraph.

For instance, the vector representing the texture feature generated by the Gabor wavelet analysis for the first selected key frame of the fifth shot has a minimum value of 27890.85 while the largest element is equal to 2873287. To confirm the effect of changing the Gabor transform parameters (the scale and orientation) upon the execution time of the feature extraction module we construct TABLE 31 that reports the total execution time of that module as a function of S and K.

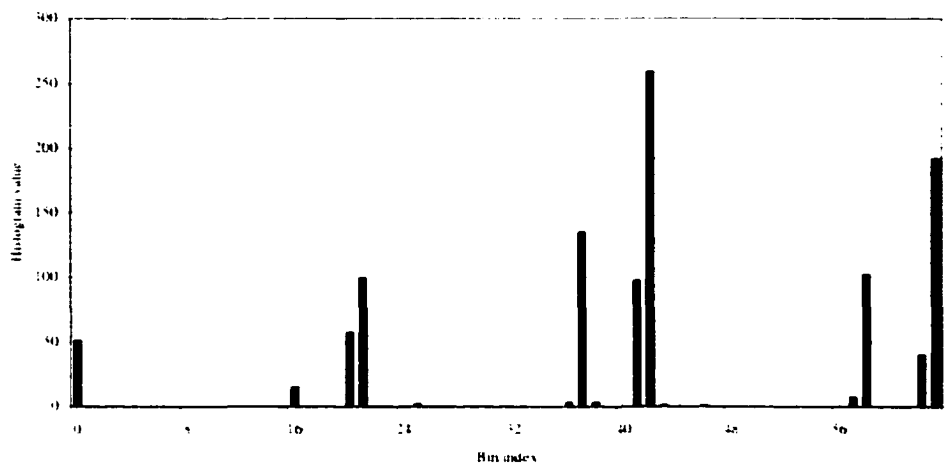

Fig. 28. The histogram of the 9th selected key frame (with index 223) of the fifth shot of the carton clip.

TABLE 30

Color Similarity Values among all Selected Key Frames for the First Three Shots of the carton Clip

	KF0	KF1	KF2	KF3	KF4	KF5	KF6
KF0	1.0	0.824248	0.875939	0.749060	0.800751	0.837406	0.787593
KF1	0.824248	1.0	0.947368	0.728383	0.760338	0.720864	0.672932
KF2	0.875939	0.947368	1.0	0.775375	0.762218	0.746240	0.695488
KF3	0.749060	0.728383	0.775375	1.0	0.671992	0.681391	0.681391
KF4	0.800751	0.760338	0.762218	0.671992	1.0	0.859022	0.843045
KF5	0.837406	0.720864	0.746240	0.681391	0.859022	1.0	0.918233
KF6	0.787593	0.672932	0.695488	0.681391	0.843045	0.918233	1.0

TABLE 31

The Execution Time of the Texture Extraction as a Function of Scales and Orientations for the First KF of the Fifth Shot of the carton Clip

S	3	3	3	4	4	4	5	5	5
K	4	5	6	4	5	6	4	5	6
Time (sec)	20.8	25.8	30.7	27.4	33.9	40.7	33.9	42.2	50.5

4.6 Conclusion

In this chapter, we introduce two techniques for indexing MPEG video data namely color and texture indexing. These techniques work upon the output produced by the algorithms developed in CHAPTER III, the set of selected key frames. This set is then passed to the color indexing module that uses the DC form of these frames (previously generated by the parsing module) to construct a color histogram for each of the selected frames. The color extraction algorithm efficiently reduces the dimension of the produced histograms while keeping a reasonably accurate representation level. These color histograms are used as feature vectors representing the selected key frames and the histogram intersection is employed to measure the similarity between two histograms.

The second indexing technique we use has the objective of extracting the texture of the selected key frames and storing the derived indexes into texture feature vectors. We employ the Gabor wavelet transform to derive these texture feature vectors. Both the color histograms and the texture vectors are stored as metadata to represent the video in any further similarity matching tasks. The performance of the proposed techniques were measured and we obtained very efficient color indexing results while the texture indexing was less efficient as a result of the time consuming algorithm used to generate these vectors. Although the texture extraction is a time consuming task, we decided to include it for now in addition to the color feature to improve our representation of video streams. The final decision regarding its inclusion is postponed until evaluating its impact on the retrieval system. The focus of the next chapter will be on how to use the stored indexes to measure the similarity of video data in an intuitive and effective way.

CHAPTER V

THE RETRIEVAL SYSTEM

The basic objective of any automated video indexing system is to provide the user with easy-to-use and effective mechanisms to access the required information. For that reason, the success of a content-based video access system is mainly measured by the effectiveness of its retrieval phase. The general query model adapted by almost all multimedia retrieval systems is the QBE (Query By Example) [96]. In this model, the user submits a query in the form of an image or a video clip (in case of a video retrieval system) and asks the system to retrieve similar data. QBE is considered to be a promising technique since it provides the user with an intuitive way of query presentation. In addition, the form of expressing a query condition is close to that of the data to be evaluated.

Upon the reception of the submitted query, the retrieval stage analyzes it to extract a set of features then performs the task of similarity matching. In the latter task, the query-extracted features are compared to the features stored into the metadata then matches are sorted and displayed back to the user based on how close a hit is to the input query. A central issue here is how the similarity matching operations are performed and based on what criteria. This central theme has a crucial impact on the effectiveness and applicability of the retrieval system.

In this chapter, we discuss the design of the retrieval stage of our VCR (Video Content-based Retrieval) system and shed the light on one of its distinguishing characteristics. Our system attempts to make its comparison decisions based on modeling the way humans perform similarity matching of video data. This is achieved by using a number of factors reflecting how humans perceive media similarity. This model for measuring the similarity of video materials overcomes the shortcomings of other approaches that overlook the very essential fact that can be stated as "similarity matching has significance only if it can emulate what humans do". Near the end of this chapter, performance analysis of the proposed retrieval phase is given reporting the success of the introduced model for measuring video data similarity.

5.1 Introduction

In the previous three chapters, the design and implementation algorithms for the three major stages of the data population phase of our VCR (Video Content-based Retrieval) system [29] were discussed. The control flow of information among those three modules is depicted in Fig. 29 below.

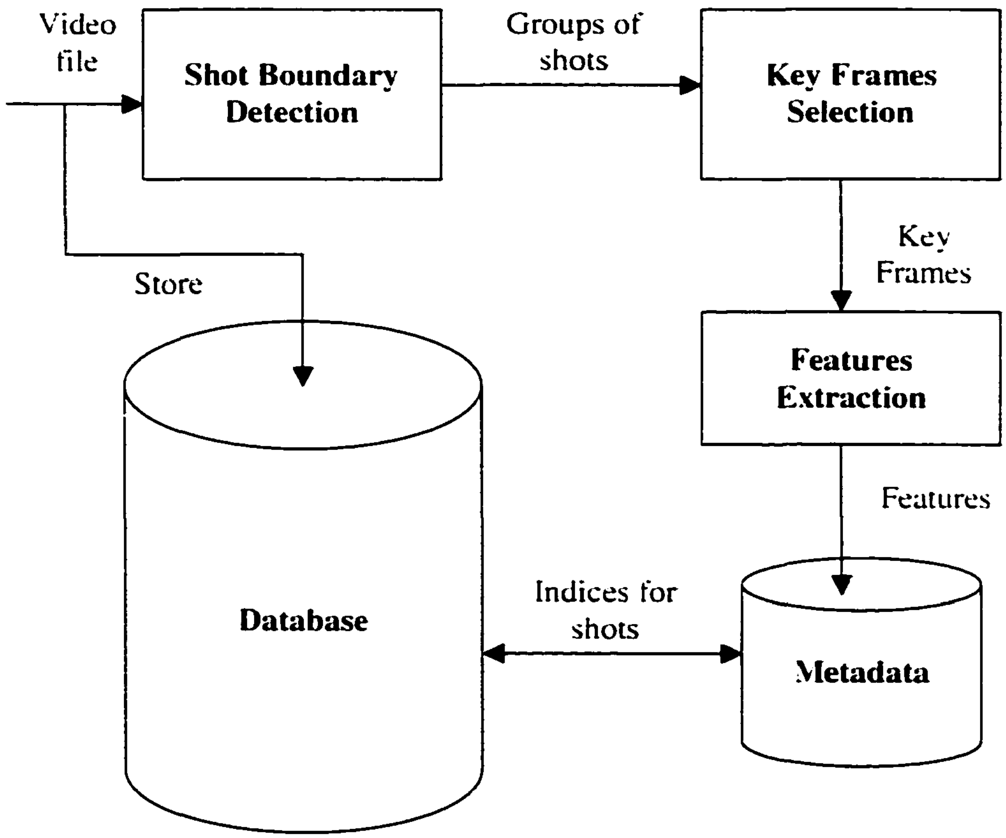

Fig. 29. The basic components of the database population phase.

As we pointed before, the database population phase is performed as an off-line activity. This process outputs a set of metadata with each element representing one of the clips in the video archive. These metadata contain a number of useful information to be used during the similarity matching process that is the focus of this chapter. Metadata of each video clip are stored in a separate directory containing the following:

- A set of the selected KFs (Key Frames) for this clip.

- A video information file storing the original clip file name, the total number of frames in the clip, the number of macroblock rows and columns.
- The full absolute path of the actual video clip is stored into a file.
- An index file that constitutes the bulk of the metadata. This file specifies the number of shots in this clip, video frame rate (in frames/second), and a series of objects each representing one video shot. The information stored in these objects was explained in the previous chapter.

The retrieval system should provide the user with an easy-to-use and effective interface in which the user can query and browse through the video library. The main scheme for submitting the query is QBE for its advantages discussed before. A high-level schematic diagram of the retrieval system is shown in Fig. 30.

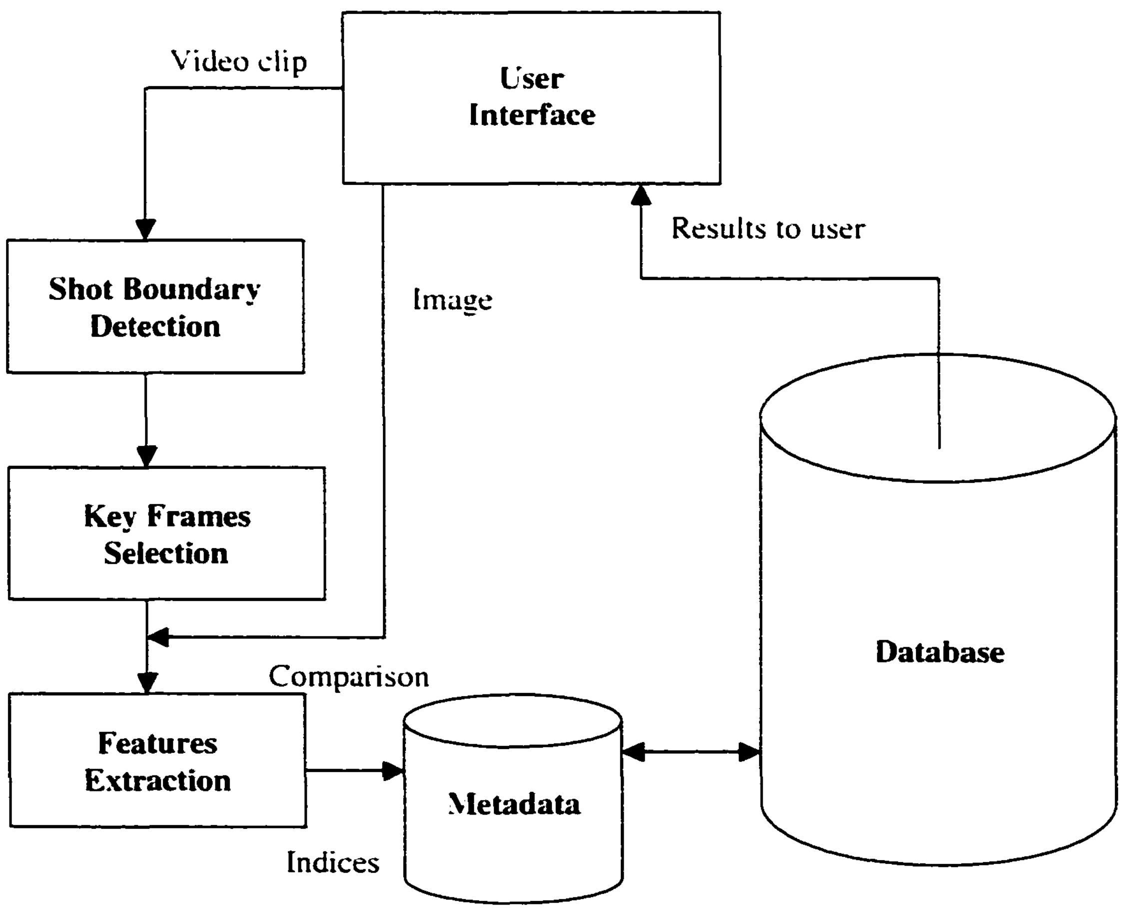

Fig. 30. Different stages of the retrieval, query processing, phase.

The steps performed in processing an input query are:

- Shot boundary detection and key frames selection are performed if the query is a video clip then features extraction is done next. otherwise features extraction is performed directly on the query image.
- Similarity comparisons among the extracted features and those stored into the metadata are performed.
- Results are returned back to the user.

5.2 Literature Review

Nowadays. the amounts of multimedia data stored on video archives are in exponential growth. This entails the real need for effective and efficient methodologies to access these data. An important lesson [10] that has been learned through the last two decades from the increasing popularity of the Internet can be stated as the worth of any information repository is only determined by how effective one can retrieve the required information. The same analogy applies to video archives thus many researchers start to be aware of the significance of providing effective tools for accessing video databases. Moreover. some of them are proposing various techniques to improve the quality. effectiveness. and robustness of the retrieval system. In the following. a quick review to these techniques is introduced.

One important aspect of multimedia retrieval systems is the browsing capability and in this context some researchers proposed the integration between the human and the computer to improve the performance of the retrieval stage. The main justification to this approach is the inaccuracies of automatic tracking algorithms. In [51] a system is proposed that allows the user to define video objects on multiple frames then the system can interpolate the video object contours in every frame. Another video browsing system is presented in [82]. [89] where comic book style summaries are used to provide fast overviews of the video content. One other prototype retrieval system that supports 3D images. videos. and music retrieval is presented in [43]. In that system each type of queries has its own processing module. for instance. image retrieval is processed using a component called ImageCompass.

A very central and significant issue in multimedia (specially video) retrieval systems is how to determine the similarity between the submitted query and the media stored into the database. For that reason, the effectiveness of the similarity matching model is a crucial factor in determining the success of a video retrieval system. A number of researchers has proposed various approaches to measure video similarity and a quick review follows. One technique was proposed in [15] to use the metadata derived from clip links and the visual content of the clip to measure video similarity. At first, an abstract form of each video clip is calculated using a random set of images then the closest frame in each video to a particular image in that set is found. The set of these closest frames is considered as a signature for that video clip. Color histograms are used to represent the visual content in conjunction with a pruning algorithm to reduce the dimensionality of the feature vector. An extension to the video signature technique just described that uses clustering of stored data is introduced in [16]. In that article, the authors stated the need for a robust clustering algorithm to offset the errors produced by random sampling of the signature set. The clustering algorithm they proposed is based upon the graph theory and it starts by measuring the edge density in each connected component and if it exceeds a certain threshold this connected component is removed from the graph. Otherwise, it starts trimming edges in decreasing order of their lengths and calculates the edge density of new clusters then proceeds. Another clustering algorithm was proposed in [49] to dynamically distinguish whether two shots are similar or not based on the current situation of shot similarity.

A different retrieval approach uses time-alignment constraints to measure the similarity and dissimilarity of temporal documents. In [92], multimedia documents are viewed as a collection of objects linked to each other through various structures including temporal, spatial, and interaction structures. The challenges in determining media similarity are due to the large number of objects, the temporal structure, and the content and role of the objects in temporal similarity. The similarity model in that work uses a highly structured class of linear constraints that is based on instant-based point formalism.

In [79], a framework is proposed to measure the video similarity. It employs different comparison resolutions for different phases of video search and uses color histograms to calculate frames similarity. Frames are aligned before applying the similarity formula based on a number of alignment constraints that are calculated using forward dynamic programming techniques. By using this method, the evaluation of video similarity becomes equivalent to finding the path with minimum cost in a lattice.

A powerful concept to improve searching multimedia databases is called relevance feedback [66], [91], [103]. In this technique, the user can associate a score to each of the returned hits and these scores are used to direct the following search phase and improve its results. One example to the relevance feedback was introduced in [103] where the authors define the relevance feedback as a biased classification problem in which there is unknown number of classes but the user is only interested in one class. They used linear/non-liner bias discriminant analysis that is a supervised learning scheme to solve the classification problem at hand.

From this quick survey of the current approaches, we can observe that an important issue has been overlooked by almost all the above techniques. This issue can be stated as "similarity matching has significance only if it can emulate what humans do" [49], [70]. Our belief in the utmost importance of the above phrase motivates us to propose a novel technique to measure the similarity of video data. This approach attempts to come up with a model to emulate the way humans perceive the similarity of video data. Our matching model will be discussed in details in Section 5.4.

5.3 The Retrieval Subsystem

The first component of the retrieval subsystem, as mentioned before, is the user interface that represents the only interacting tool between users and the retrieval subsystem. The main functions of that interface are:

- To allow users to issue queries in an easy and intuitive way.
- To provide users with an effective visualization tool to enable them to browse through the results of their queries and to select any of the output clips to be played or to be used as a new search example.

The technique used to formulate a query is the QBE as mentioned before in which the user requires the system to retrieve similar items to the query example. The only assumption needed in order to use QBE is that the user has an image (or a video clip) to be used as a query. One issue here is the case where the user does not have a query example so the assumption required by QBE is not satisfied. To overcome this issue, the user can start by browsing the database and then selecting an item to be used as a query example.

A typical scenario for a search session is as follows. The user starts by supplying the system with a query (an image or a video clip) and requests the system to look for similar items. The system then searches all the available files and displays the results as key frames associated with file names of the clips containing these frames. The order of the displayed results will depend on the degree of similarity of each item with the query example. The user can then browse through the displayed results and choose any of them to start playing back at that point or as a new query.

Here, a very important issue arises that is the efficiency of the searching process. If the above algorithm is implemented as described, the time to retrieve the required video clip will be directly proportional to the number of clips stored into the database. In our research, we are addressing medium to large video archives and a direct linear search will result in very slow response time that prevents any type of interactivity. This will preclude the use of the proposed system as on-line video search engine. To tackle this issue, we proposed a new scheme for improving the efficiency and scalability properties of the search process. The idea is to use a two-stage search procedure, these stages are:

- The filtering stage: Where the accuracy of the matching operation is not high. At the end of this stage most of irrelevant items are excluded and the search space will be much more confined.
- The actual comparison stage: Where detailed matching operations are performed to find out relevant video data. These matching operations are performed over the results of the previous filtering stage.

5.3.1 The Filtering Algorithm

The algorithm we proposed to implement the filtering stage is a simple but effective way to reduce the size of the search space. It works only if the input query is a video clip and it does not apply to image queries. The basic concept is to derive a signature for each shot and compare these signatures (with some tolerance) at the beginning of the search process. If the signature of a query shot is approximately similar to a metadata shot signature, the database video clip containing this shot is considered relevant to that query shot. That database clip is now a candidate for the second search stage. One assumption we postulate here is that if one shot of a database clip is relevant to one of the query shots, the whole database clip will be elected as one input to the accurate comparison stage, the second one. The signature we used to represent each shot is the relative distance between selected key frames. For instance, shot 0 of the action-movie clip has three key frames with indexes 0, 31, 45 (see TABLE 19) so that its signature is a vector with two values (31, 14).

The filtering algorithm works as follows:

- Calculate the signature of the query shot and the database shot.
- If the length of the query signature is greater than the length of the current database shot signature, then the two shots are not relevant.
- If the two shots have the same signature length, check to see if the two signatures are approximately similar and if so the two shots are relevant, otherwise they are not relevant.
- If the length of the database shot signature is larger than that of the query, check if the signature of the query shot is a subset of this database shot signature and if so the two shots are relevant, otherwise they are not.

The above-proposed filtering algorithm is an effective tool to reduce the size of the search space and facilitate interactive query processing. The only problem we encountered while evaluating its performance occurs with queries that are edited versions of some database videos. In such cases, the signature of the edited clip is different from the signature of the original clip stored into the database. This may cause our filtering stage to exclude some database video clips albeit they might be relevant to the input query. To avoid such a situation, the system gives the user the ability to disable the filtering stage and this way the user can compromise the increase in the response time with the accuracy of the returned results. By giving the user such a control, the system becomes flexible enough to accommodate users with various requirements. The effectiveness of the filtering stage in speeding up the search process will be evaluated in Section 5.5.4.

5.3.2 The User Interface and the Browsing Environment

We designed an easy-to-use user interface with two major goals in mind. The first one is to allow the user to navigate the video archive (or any other place in the storage devices) to select a query. Fig. 31 shows the part of the interface that is created when the user press the browse button. The stand-alone frame shown in Fig. 31 enables the user to select a query and automatically disappear when the selection is done. Our second goal in designing the interface is to provide the user with an effective visualization tool to browse through the returned search results. A snapshot of part of the user interface displaying some of the retrieved clips is depicted in Fig. 33.

The proposed browsing environment does worth some explanation at this point. We propose a visual browsing approach where each shot is represented (in the display interface) by its first key frame. The initial display of a set of shots will be their representative key frames (one for each shot). The results of a query will be displayed using the key frames organization just described and the same structure will allow the user to navigate the returned hits. One important point that is related to the browsing system is the ability given to the user to further refine the results of his search. So after he/she browses through the results produced by the system as a response to the first submitted query, one of the displayed items can be selected as a new query. This process can be repeated many times with the ultimate goal of improving the accuracy of the search results. In this scenario both human experiences and the powerful features of computers are integrated in order to improve the performance of the retrieval system.

5.4 The Similarity Matching Model

The outputs of the filtering stage will be used as inputs to the second search stage that performs accurate similarity matching based on the contents of the query and those stored into the database. Here comes up one of our main contribution in the retrieval system in which a viable and distinguishable approach for performing the similarity matching phase, that constitutes the bulk of the search procedure, is proposed. This technique attempts to model the human perceptual way of judging video similarities. Although a faithful implementation of the human ability is beyond any proposed artificially automated system, a system that can partly model this ability can be proposed. The details of this similarity model are the topic of this section.

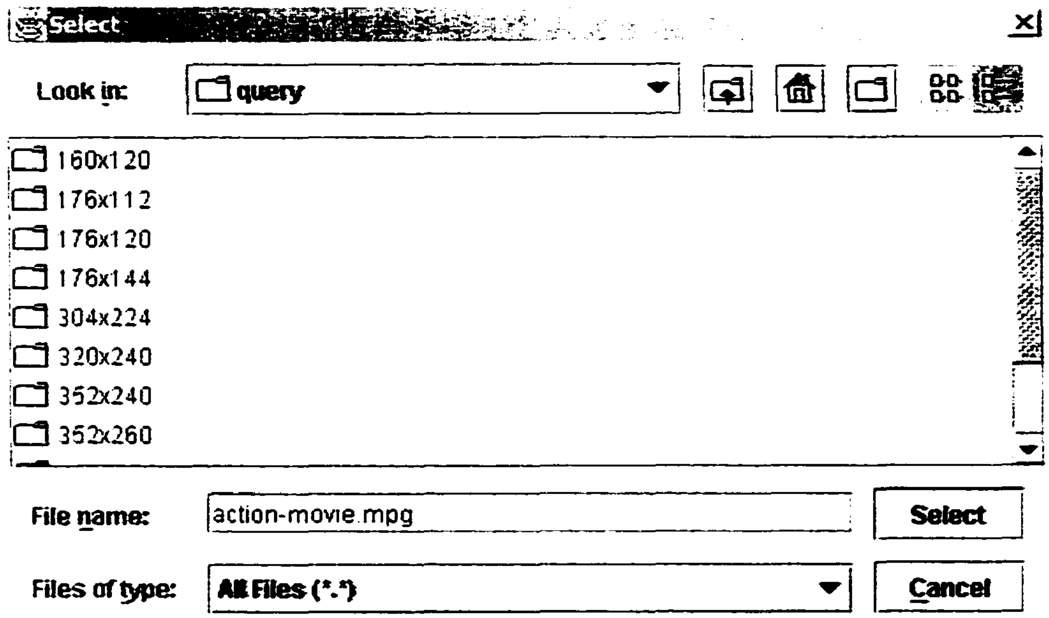

Fig. 31. A separate frame of the interface to enable the selection of the query example.

Before discussing the details of the proposed similarity model, let us define the three types of input queries that the retrieval system can receive, these are:

- A query image: In this case, neither the filtering stage nor the human-based similarity factors we propose are applicable because the image has no temporal dimension. The system extracts the color and texture features from an input image, in JPEG format [60], and compares these features with all the features vectors stored into the metadata. The hits are sorted and returned to the user based on how similar they are to the input query. Again, for the lack of time dimension, there is no meaning for similar shots in this context. Fig. 32 illustrates an example of an unseen image query while Fig. 33 shows the search results returned as a response of submitting the image in Fig. 32 as a query to the system.
- A one-shot video clip: This is a special case of the next type and in these two types both the filtering stage and the similarity matching model are applicable. The returned search results will be the overall similarity hits, the most similar

video clips to the input query as a whole. In addition, clips that are similar to the first shot will be retrieved too but in this particular case the two sets will be identical.

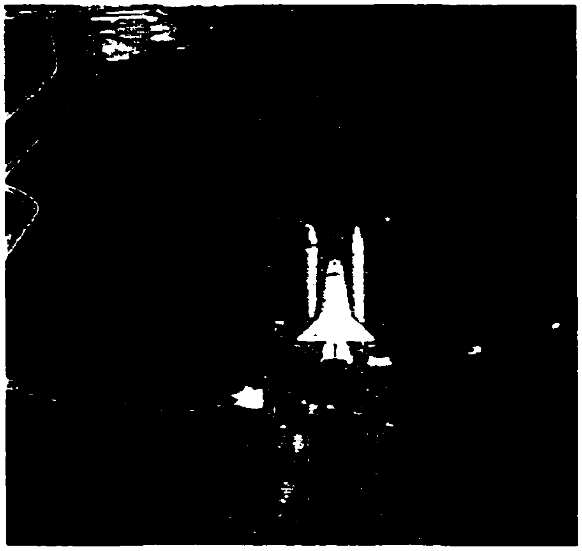

Fig. 32. An example JPEG image query.

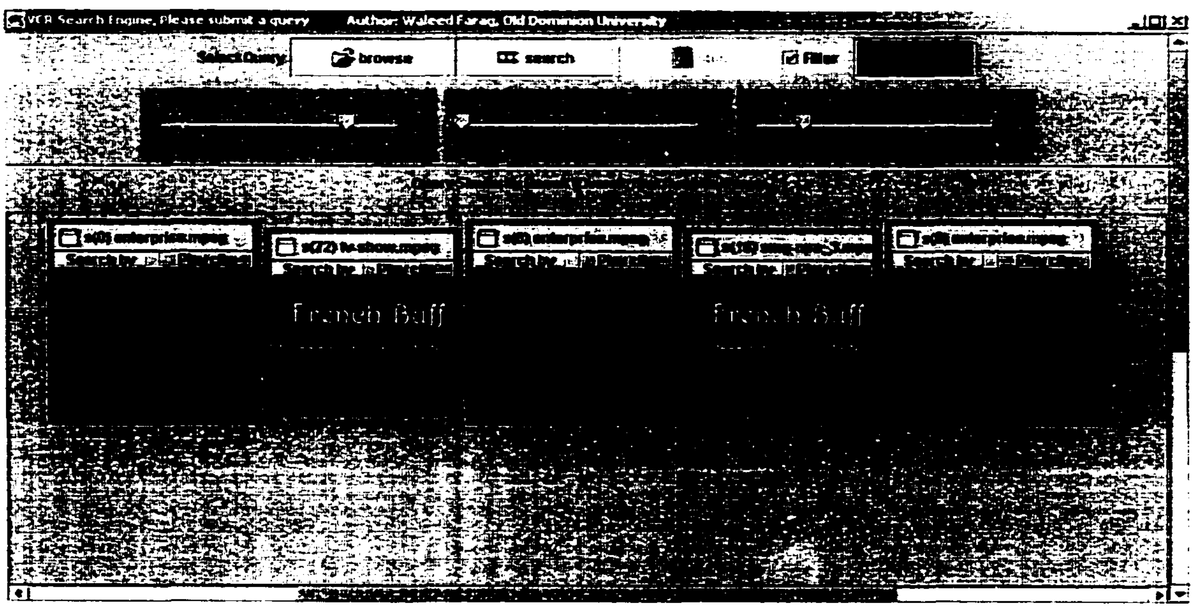

Fig. 33. The results obtained by submitted the image in Fig. 32.

- Multi-shot video clip: This is the general case where the input is a normal video clip that has more that one shot. The retrieval system starts by analyzing the input clip, the same way performed during the data population phase, and producing an index file (in addition to other metadata information) for this particular query. Then the filtering stage will be applied producing a group of candidate clips for the accurate comparison stage. The latter stage measures the similarity between shots based on the shot content (using the color and the texture features as described in the indexing chapter). In addition to the visual content, a number of other factors aiming at modeling the human perception are also involved in determining the final similarity values. Fig. 34 illustrates part of the interface obtained as a result of submitting the clip "action-movie" as a query example.

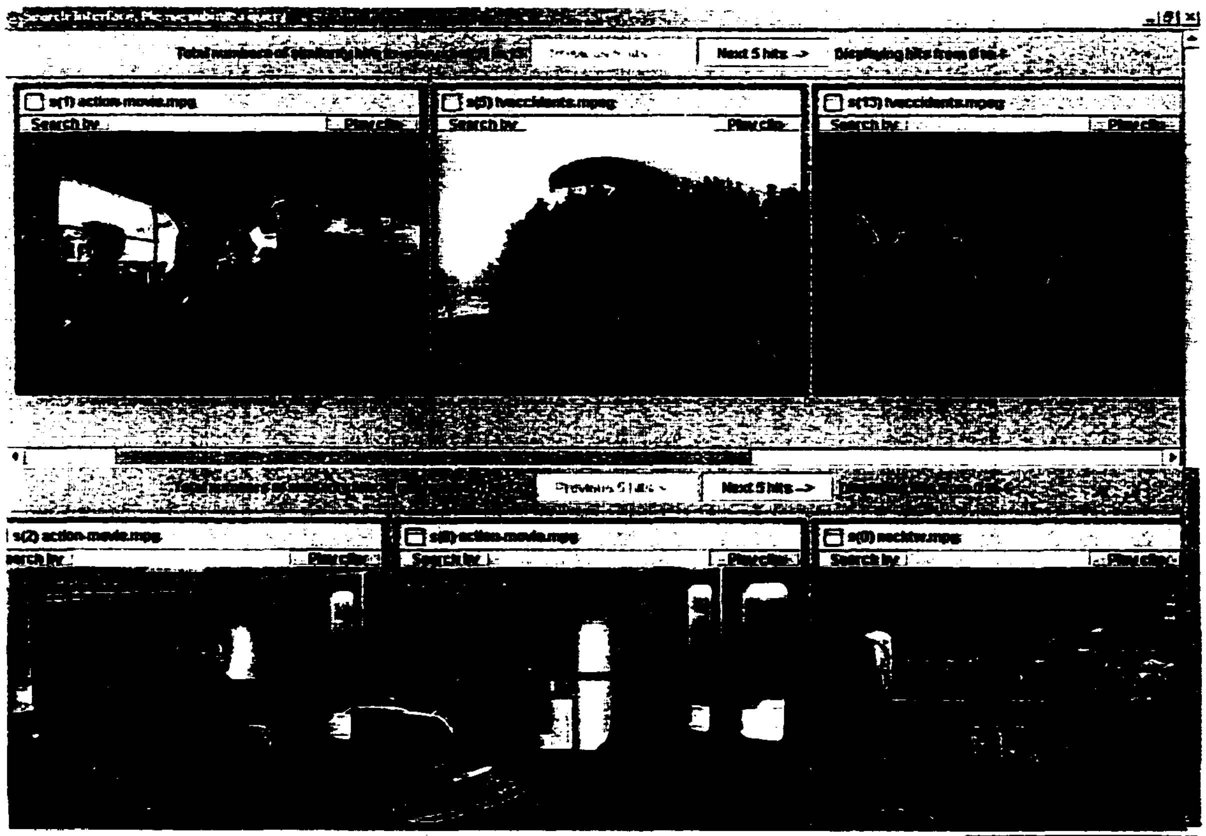

Fig. 34. Part of the search results obtained by submitted the action-movie clip as query.

Starting from this point, our attention will be focused on input video queries, the last two types, because the assumptions and techniques that will be discussed cannot be applied to image queries. In order to lay the foundation of our similarity matching model, a number of assumptions will be listed first, these are:

- The similarity of video data is based on the similarity of their constituent shots. That is to say, we measure the similarity between shots and use these similarity values (along with other measures) to determine the overall clip-to-clip similarity.
- If the signature of the query shot is longer (in length) than the signature of a database shot then the two shots are not relevant.
- If only one shot of the input query is relevant to a shot in one of the database clips, the whole database clip is considered relevant to the input query.
- The query clip is usually much smaller than the average length of the database clips otherwise the processing time at query invocation will be relatively large limiting the interactivity of the system.

The results of submitting a video clip as a search example is divided into two levels. The first one is the query overall similarity level which lists similar database clips sorted according to their degree of similarity to the input query. The second level gives a finer level of granularity in measuring the similarity of video data. In this level, the system displays a list of similar shots to each shot of the input query and this gives the user a much more detailed results based on the similarity of individual shots. It has been reported that humans are fickle in their decisions from time to time [62]. Thus this finer level of measuring the similarity plays a vital role in improving the retrieval performance and helps the user in determine what they really interested in. To illustrate the benefits of such approach we use the following example. Suppose a user submits a two-shot clip as a query and he is more interested in the content of the second shot of this query than that of the first shot. The system in response will retrieve a number of database clips that is similar to the input query as a whole. In addition, the system will return a list of similar database shots to each shot of the query. There is a high probability that overall similar clips will not be as important to the user as those shots returned by the system as similar to the second shot alone.

5.4.1 Shot Similarity Definition

In the above stated assumptions, we define video data similarity based on the similarity of individual shots, so let us go further and explain our definition of shot similarity. A shot is a sequence of frames so we need to formulate first frames similarity.

In our model, the similarity between two video frames is defined based on their visual content where color and texture are used as visual content representative features. Color similarity is measured using the normalized histogram intersection while texture similarity is calculated using a Gabor wavelet transform technique discussed in the indexing chapter. We use equation (38) to measure the overall similarity between two frames *f1* and *f2* where S_c (color similarity) is defined in equation (39) and S_t (texture similarity) is defined in equation (36) in CHAPTER IV.

$$Sim(f1,f2) = 0.5 * S_c + 0.5 * S_t \tag{38}$$

$$S_c = \frac{\sum_{i=1}^{n} Min(H_{f1}(i), H_{f2}(i))}{\sum_{i=1}^{n} H_{f1}(i)} \tag{39}$$

Equal weights have been used in equation (38) to give equal emphasis to both the color and the texture features in determining frames similarity. One issue we need to highlight here regarding measuring the color similarity is the case of different dimension frames. We use various video clips in our database that vary in frame rate, length (number of frames), dimension (width and height in pixels), and other characteristics. When calculating the color histogram intersection the variation of frames dimension should be considered. To solve this issue, the query frame histogram is scaled before calculating its normalized histogram intersection using equation (40), where *size(fi)* is the product of the width and the height of frame i.

$$H'_{f1}(i) = \frac{size(f2)}{size(f1)} * H_{f1}(i) \tag{40}$$

Suppose we have two shots *S1* and *S2* each has *n1* and *n2* frames respectively. An intuitive way to measure the similarity between these shots is to measure the similarity between every frame in *S1* with every frame in *S2* and form what we called the similarity matrix that has a dimension of *n1Xn2*. For the ith row of the similarity matrix the largest element value represents the closest frame in shot *S2* that is most similar to the ith frame in shot *S1*. Similarly, the largest element value in the jth column represents the closest frame in shot *S1* that is most similar to the jth frame in shot *S2*. By calculating the summation of these closest frames over all the rows and columns of the similarity matrix and dividing that summation by the total number of rows and columns in that matrix, we come up with a definition of the similarity between two video shots. Equation (41) is the mathematical definition of our shot similarity criterion.

$$Sim(S1,S2) = \frac{\sum_{i=1}^{n1} Max_{row(i)}(S_{i,j}) + \sum_{j=1}^{n2} Max_{col(j)}(S_{i,j})}{n1+n2} \quad (41)$$

Where

$Max_{row(i)}(S_{i,j})$: is the element with the maximum value in the ith row.

$Max_{col(j)}(S_{i,j})$: is the element with the maximum value in the jth column.

$n1$: is the number of rows in the similarity matrix.

$n2$: is the number of columns in the similarity matrix.

One can remark that direct application of equation (41) is practically infeasible due to the large number of frames into video shots. For instance, a 10-second shot has 300 frames (assuming 30 frame/seconds rate). To avoid this computational complexity, we need to use an abstract form of the video shot to measure the similarity upon and a perfect candidate, we already used before, is the set of key frames. Thus, the set of key frames for each shot will be employed in calculating the shot similarity using equation (41) and this results in a dramatic speed up in measuring shot similarity. For example, we got around 400 times improvement in computation time when applying equation (41) upon key frames to calculate the similarity between two 60-frame shots with 3 key frames each.

5.4.2 Human-based Similarity Factors

After we introduce our assumptions and expound the definition of shot similarity, the background has been established to explain our similarity matching model. It has been mentioned before that our similarity model attempts to emulate the way humans perceive the similarity of video material. This was achieved by integrating into the similarity measuring formula a number of factors [49] that most probably humans use to perceive video similarity. These factors are:

- **The visual similarity**: Usually humans determine the similarity of video data based on their visual characteristics such as color, texture, shape, etc. For instance, two images with the same colors are usually judged as being similar.

Our similarity model reflects this factor by using the color and texture of key frames as low-level features describing the visual content of the video/image data.

- **The rate of playing the video**: Humans tend also to be affected by the rate at which frames are displayed and they use this factor in determining the degree of video similarity. For example, two videos that have the same frame rate have a high probability of being judged as similar clips by a human observer. To take this factor into consideration, we included it in the final similarity calculation as will be explained later.
- **The time period of the shot**: The more the periods of video shots coincide, the more they are similar to human perception. We consider the duration factor in our similarity measuring formula to reflect the effect of this parameter.
- **The order of the shots in a video clip**: Humans always give higher similarity scores to video clips that have the same ordering of corresponding shots. Thus, our model includes the ordering factor in determining the similarity of video data.

As a result of the above discussion, we need to integrate these human-based similarity factors into the similarity calculation. Consequently, we include all these factors in one similarity matching formula given in equation (42).

$$Sim(S1,S2) = W_1 * S_V + W_2 * D_R + W_3 * F_R \tag{42}$$

$$D_R = 1 - \frac{|S1(duration) - S2(duration)|}{Max(S1(duration), S2(duration))} \tag{43}$$

$$F_R = 1 - \frac{|S1(rate) - S2(rate)|}{Max(S1(rate), S2(rate))} \tag{44}$$

Where

S_V: The visual similarity calculated using equation (41).

D_R: Shot duration ratio defined in equation (43).

F_R: Video frame rate ratio defined in equation (44).

$Si(duration)$: Time duration of the ith shot.

$Si(rate)$: Frame rate of the ith shot.

W_1, W_2, *and* W_3: Relative weights used as parameters to the algorithm.

There are three parameter weights in equation (42), namely, W_1, W_2, *and* W_3 that give the system indication on how important a factor is over the others. For example, stressing the importance of the visual similarity factor is achieved by increasing the value of its associated weight (W_1). The question now is "how to select the values of these weights to best reflect what humans do in measuring video similarity". Our first trial was to test various randomly selected combinations of these weight values to choose the best set of weights. A second idea is to use an optimization algorithm such as the Genetic Algorithm [19], [33] to optimize the values of these weights and produce the best combination set. In the first case, we need a measure of similarity that should always work with various users while in the second one, an evaluation function is required to give merits to individuals who perform better than other members of the population. Selecting fixed similarity criteria or evaluation function(s) by one or even few individuals do not necessary reflect the opinion of other users of the system. The last statement is supported by the fact that a single individual is even fickle with respect to his/her opinions from time to time [62].

For the reasons mentioned in the previous paragraph, we found it too restrictive to select these parameters for the user in advance (either by using a pre-selected set of values or by using the genetic algorithm to select the best combination set). Consequently, we decided to give the user the ability to express his/her real need by allowing these parameters to be adjusted by the user before submitting the query. Three easy-to-use sliders, one for each weight parameter, are supplied into the interface (see Fig. 33) to enable the user to select the values of these weights according to his preferences. For instance, if a user is interested in retrieving video clips that are mainly similar in visual content to the given query, he should select a large value for W_1 and small values (or even zeros) for the other two weights. Another example is a user that wants to retrieve videos that are similar in content and at the same time have similar frame rates to the query. In that case, the user should stress both W_1 and W_3. If the user is not decided or does not know what values of weights are better, then the system will select default values for these weights that stress visual similarity in calculating the similarity formula. This way, the system exhibits a great degree of flexibility to various situations and users opinions and flavors. A few clicks before issuing a new query is all what it takes to direct the system to what the user actually wants. The performance of the proposed similarity model will be evaluated at the end of this chapter.

To reflect the effect of the order factor, the overall similarity level should check if the shots in the currently investigated database clip have the same temporal order as the shots in the query clip. To achieve this goal, we propose the following algorithm.

- *Select the first five hits (if any) that are similar to the first query shot (five is a parameter of the algorithm) and assign this value to the variable size.*
- *For i = 0 To i < size {*

  For j = 1 To j < Number of shots in the query {

  For k = 0 To k < Number of database shots similar to this query shot {

  If (the two shots belong to the same clip and in consecutive order) {

  Check the next shot

  }

  } // repeat for each k

  } // repeat for each j

  If (the query shots temporal order is respected by this database clip) {

  The current database clip is chosen in the overall similarity level

  }

  } // repeat for each i

Although the above algorithm restricts the candidate of the overall similarity set to clips that have the same temporal order of shots as the query clip, we still have a finer level of similarity that is based on individual query shots. This level can capture other aspects of similarity as discussed before.

5.5 Performance Evaluation

Evaluating video retrieval systems is still an open area of research [12] and this problem arises due to the lack of standard benchmarks to measure system performance. To evaluate the performance of our system we start by using the standard precision-recall metrics borrowed from the information retrieval field. Precision and recall can be defined by equations (45) and (46) respectively based on the symbols listed in TABLE 32 [37].

$$R = \frac{A}{A + C} \tag{45}$$

$$P = \frac{A}{A+B} \quad (46)$$

Where

R: The recall value.

P: The precision value.

TABLE 32

Retrieval Symbols Definition Table

	Relevant	Not relevant
Retrieved	A (correctly retrieved)	B (incorrectly retrieved)
Not retrieved	C (missed)	D (correctly rejected)

Before discussing the details of the experimental results, we need to highlight one important issue that is the relatively long time taken to calculate the texture features we experienced. This time represents a major portion of the total execution time of our system. In spite of the fact that our attempt to reduce this time by using a native code for this part of the system relatively succeeds, the texture algorithm execution time still represents a significant portion. This problem is due to the fact that texture calculation algorithms are in general very computationally demanding. This problem does not represent a serious issue in the database population phase that is performed as an off-line activity. On the other hand, the user of the retrieval system expects a reasonable response time for his/her query. In the latter stage, the system has to analyze the input query clip, a time consuming task by itself, extract the color and the texture features, then finally calculate the similarity values. All these delays contribute to the total response time experienced by the user of the system. To keep the interactivity property of our system, we decided to disable the calculation of the texture features at query time and just use the color as a visual content descriptor. All of the following experimental results use only the color feature in calculating the visual similarity of video data.

It is important to note that the values of both recall and precision depend on the number of allowed shots to be retrieved as a response to the query. Thus, during the following experimental results the allowed shots number will be changed to measure the values of recall and precision. One parameter that needs to be determined before calculating recall and precision is the set of relevant shots to a particular query known as the ground truth. This ground truth set is determined manually, by a human observer, before submitting a query to the system. Because of being a manual process, the selection of the ground truth is performed on a subset of our database. In Section 5.5.1 and Section 5.5.2 the performance of the system will be measured for queries that belong to the database and those that are not respectively.

5.5.1 Experimental Results Using Queries from the Database

We start by choosing the ground truth for five shots with diverse contents that are part of the database, these shots belong to five database clips called action-movie, dbvath-qcif, racing-boats, ads, and smg-npa-3. The manually obtained ground truth sets for each of the above five shots are shown in TABLE 33 through TABLE 37 respectively.

TABLE 33

Ground Truth of Shot 2 of action-movie Clip

Relevant clip name	Relevant shot indexes
action-movie	0, 1, 2
necktw	0
conf-discussion	18
tv-show	18, 36, 39, 50, 52, 55, 58, 60, 64, 66
tvaccidents	11
srose-p4	10

TABLE 34

Ground Truth of Shot 0 of dbvath_qcif Clip

Relevant clip name	Relevant shot indexes
dbvath_qcif	0, 1
dbvath_cif	0, 1
tvaccidents	3, 4
racing-boats	0, 1, 2, 3, 4, 5
speed167	0, 3
tennis1	0
documentary	2, 4
carton	8
v-hi	1
srose-p2	13

TABLE 35

Ground Truth of Shot 0 of racing-boats Clip

Relevant clip name	Relevant shot indexes
racing-boats	0, 1
speed167	2, 3
tvaccidents	1, 3, 6, 9, 14
hoey_v_kill	0
sprint	1
adecco	0
documentary	7
crawle	1
lions	0

TABLE 36

Ground Truth of Shot 1 of ads Clip

Relevant clip name	Relevant shot indexes
ads	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
srose-p2	5, 20
carton	2, 6
tv-show	12, 14, 41, 43, 45
news-cast	0

TABLE 37

Ground Truth of Shot 9 of smg-npa-3 Clip

Relevant clip name	Relevant shot indexes
smg-npa-3	2, 5, 9, 13
tv-show	2, 6, 28, 30, 32, 34, 37, 51, 53, 59, 61, 63, 65, 67, 69, 71

Fig. 35 shows part of the retrieval interface displaying the first three hits obtained as a response to submitting the first shot (shot 0) of the racing-boats as a query. The obvious relevance of the returned shots to the query is quite evident in this figure. Another example is shown in Fig. 36 that displays the first twenty hits resulted from submitting shot 9 of smg-npa-3 clip to the system. One can observe that all the retuned shots are shots that contain the same character pictured in the query shot and that the first hit, the most similar to the query, is the query shot itself. Moreover, some of the returned shots belong to the same clip in which the query shot is part of while the others are shots from a different clip. The performance of the retrieval system is very good in this query in which all the relevant shots have been retrieved without any misses or false alarms.

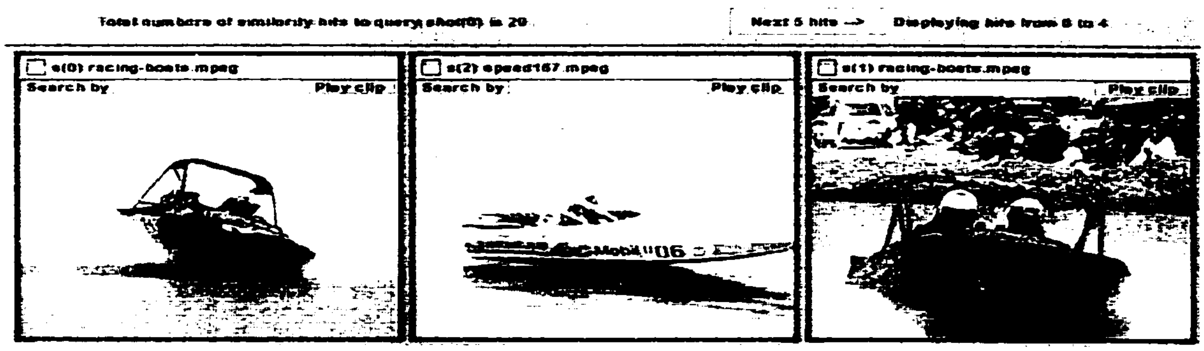

Fig. 35. The first three hits for shot 0 of the racing-boats clip.

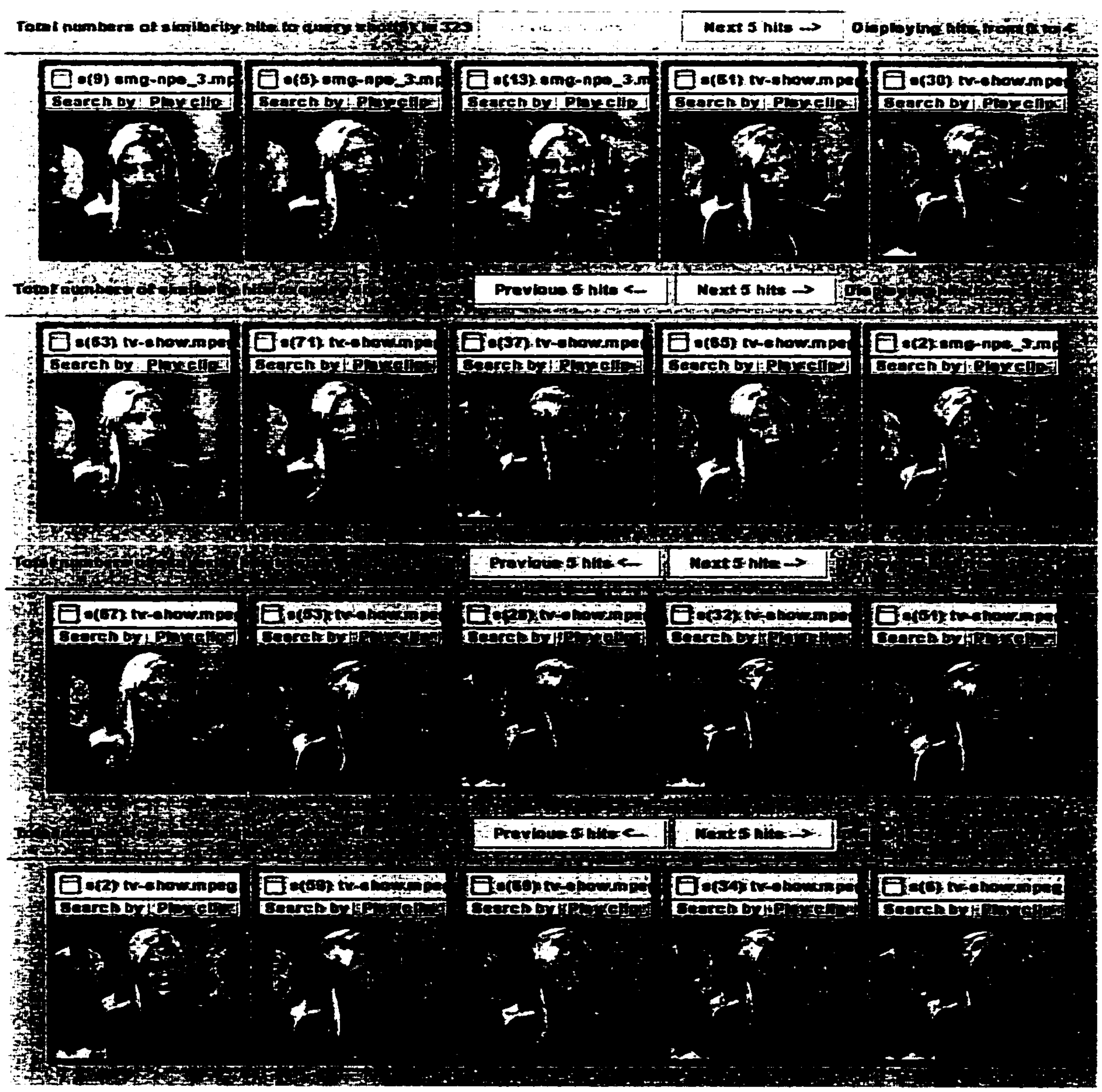

Fig. 36. The first twenty hits for shot 9 of the smg-npa-3 clip.

To measure the recall and precision of the system in response to submitting each of the above five shots as a query example, we change the returned shots number from 5 to 20 and calculate both metrics. The results of this procedure are shown in TABLE 38 through TABLE 42 for each of these five seen query shots respectively.

TABLE 38

Recall and Precision as Functions of Returned Shots Number for Shot 2 of action-movie

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	12	0.294	1.0
10	9	1	8	0.529	0.9
15	13	2	4	0.761	0.867
20	17	3	0	1.0	0.85

TABLE 39

Recall and Precision as Functions of Returned Shots Number for Shot 0 of dbvath_qcif

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	15	0.25	1.0
10	10	0	10	0.5	1.0
15	15	0	5	0.75	1.0
20	20	0	0	1.0	1.0

TABLE 40

Recall and Precision as Functions of Returned Shots Number for Shot 0 of racing-boats

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	10	0.33	1.0
10	7	3	8	0.467	0.7
15	11	4	4	0.733	0.733
20	15	5	0	1.0	0.75

TABLE 41

Recall and Precision as Functions of Returned Shots Number for Shot 1 of ads

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	15	0.25	1.0
10	10	0	10	0.5	1.0
15	15	0	5	0.75	1.0
20	20	0	0	1.0	1.0

TABLE 42

Recall and Precision as Functions of Returned Shots Number for Shot 9 of smg-npa-3

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	15	0.25	1.0
10	10	0	10	0.5	1.0
15	15	0	5	0.75	1.0
20	20	0	0	1.0	1.0

To better visualize the data in TABLE 38 through TABLE 42, the data has been plotted in Fig. 37, the recall curve, and Fig. 38, the precision curve, shown below where the five shots are named a, b, c, d, and e respectively. One can observe the predicted increase in the recall value with the increase in the number of returned shots and as soon as the latter number exceeds the number of relevant shots (see the ground truth tables) the recall value reaches 1.0 for all the shots considered in these figures. Another remark is that the recall curves for shots b, d, and e coincide showing that the system exhibits relatively steady performance over different query contents. Similarly, we can remark that the precision of the system in Fig. 38 is also very good and in some cases its value is one regardless the number of the returned shot, an indication of the system accuracy.

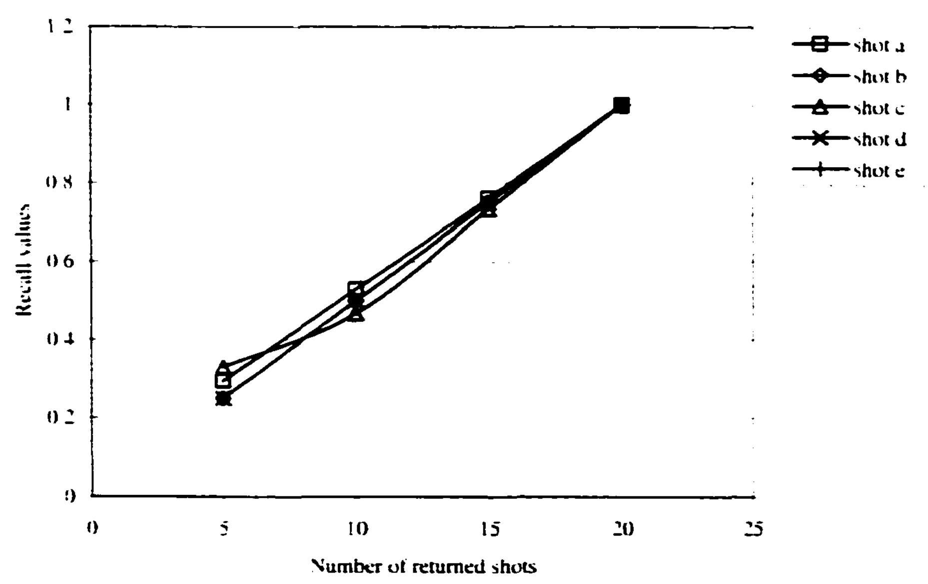

Fig. 37. Recall values for the five seen query shots.

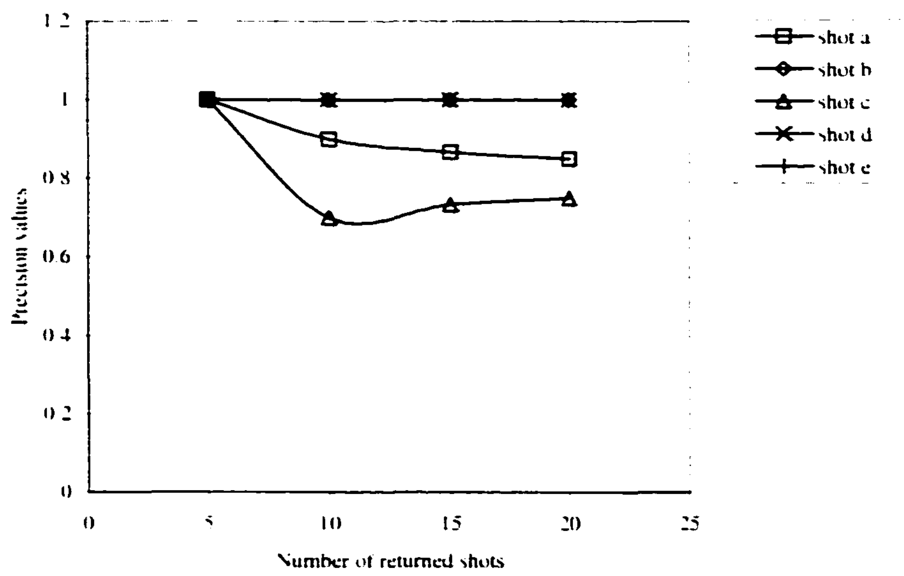

Fig. 38. Precision values for the five seen query shots.

As mentioned before, both recall and precision depend of the number of returned shots. To increase recall, more shots have to be retrieved, which will in general result in a decreased precision. To investigate the interrelation between recall and precision their values obtained above in TABLE 38 through TABLE 42 are averaged and the results are listed in TABLE 43. A graph of their relation is also shown in Fig. 39 that indicates a very good performance achieved by our system. At small number of returned shots the recall value was small while the precision value was very good. Increasing the number of returned clips increases the recall until it reaches one, at the same time the value of the precision does not degraded very much but the curve almost dwells at a precision value of 0.92. This way, the system provides a very good tradeoff between recall and precision.

TABLE 43

Average Values of Recall and Precision for the Five Seen Query Shots

Number of returned shots allowed	Recall	Precision
5	0.2748	1.0
10	0.4992	0.92
15	0.7488	0.92
20	1.0	0.92

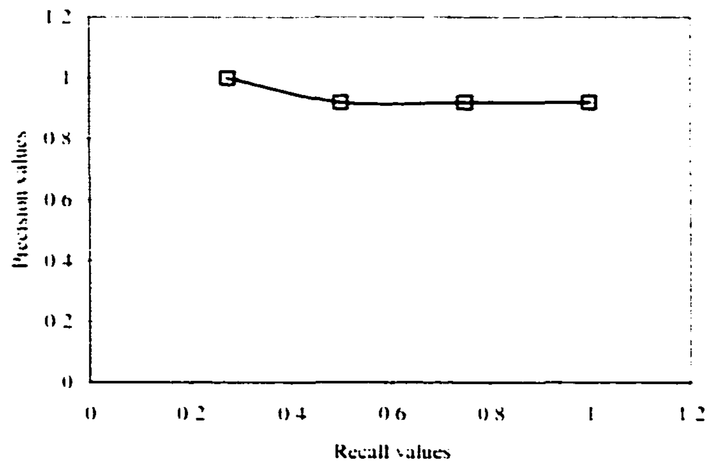

Fig. 39. Recall versus precision for the five seen query shots.

5.5.2 Experimental Results Using Unseen Queries

In Section 5.5.1, the outstanding performance of our VCR system in retrieving similar video clips to an input query has been demonstrated. Here, the generalization ability of the system will be tested by submitting queries that are not part of the video database and have never been seen by the system before. The similar trend of using the recall and precision metrics in measuring the performance will be followed as in Section 5.5.1. First, three shots out of three unseen clips are selected one from each clip. The names of these three clips are can-roll, two-goal, and thing-learn. A manual formation of the ground truth was performed with the results listed in TABLE 44 through TABLE 46 for each of the above three unseen shots respectively.

TABLE 44

Ground Truth of Shot 0 of can-roll Clip

Relevant clip name	Relevant shot indexes
reach1	0
reach2	0
lightbulb	0
thing	0
tvaccidents	12
two-armed	0
adver-sound	0, 3
smg-npa-3	7, 15

TABLE 45

Ground Truth of Shot 0 of two-goal Clip

Relevant clip name	Relevant shot indexes
adver-sound	3
reach1	0
reach2	0
thing-gaits	0
smg-npa-3	4, 7
ads	1, 3

TABLE 46

Ground Truth of Shot 0 of thing-learn Clip

Relevant clip name	Relevant shot indexes
thing	0
thing-gaits	0
thing-quest	0
adver-sound	3
enterprise	0
smg-npa-3	4, 7
ads	1

Fig. 40 is a snap shot showing the first key frame of shot 0 of the can-roll clip while Fig. 41 represents part of the retrieval interface displaying the first three hits obtained as a result of submitting that query shot. Although, the system did not see the query shot before it searches its database for similar items and returns a number of shots with almost identical content (robot arms) as the query input. The generalization ability of our system is evident in this figure.

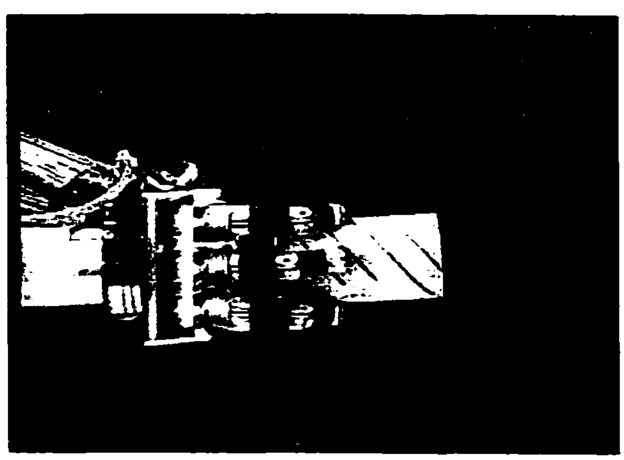

Fig. 40. The first key frame of shot 0 of the can-roll clip.

Fig. 41. The first three hits for shot 0 of the can-roll clip.

TABLE 47. TABLE 48. and TABLE 49 show the recall and precision of the system in response to submitting each of the above three shots as a query example. we change the returned shots number from 5 to 15 and calculate both metrics. TABLE 50 depicts the average values of recall and precision listed in TABLE 47 through TABLE 49.

TABLE 47

Recall and Precision as Functions of Returned Shots Number for Shot 0 of can-roll

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	5	0.5	1.0
10	8	2	2	0.8	0.8
15	10	5	0	1.0	0.667

TABLE 48

Recall and Precision as Functions of Returned Shots Number for Shot 0 of two-goal

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	3	0.625	1.0
10	7	3	1	0.875	0.7
15	8	7	0	1.0	0.533

TABLE 49

Recall and Precision as Functions of Returned Shots Number for Shot 0 of thing-learn

Number of returned shots allowed	A	B	C	Recall	Precision
5	2	3	6	0.25	0.4
10	6	4	2	0.75	0.6
15	8	7	0	1.0	0.533

TABLE 50

Average Values of Recall and Precision for the Three Unseen Query Shots

Number of returned shots allowed	Recall	Precision
5	0.458	0.8
10	0.81	0.7
15	1.0	0.576

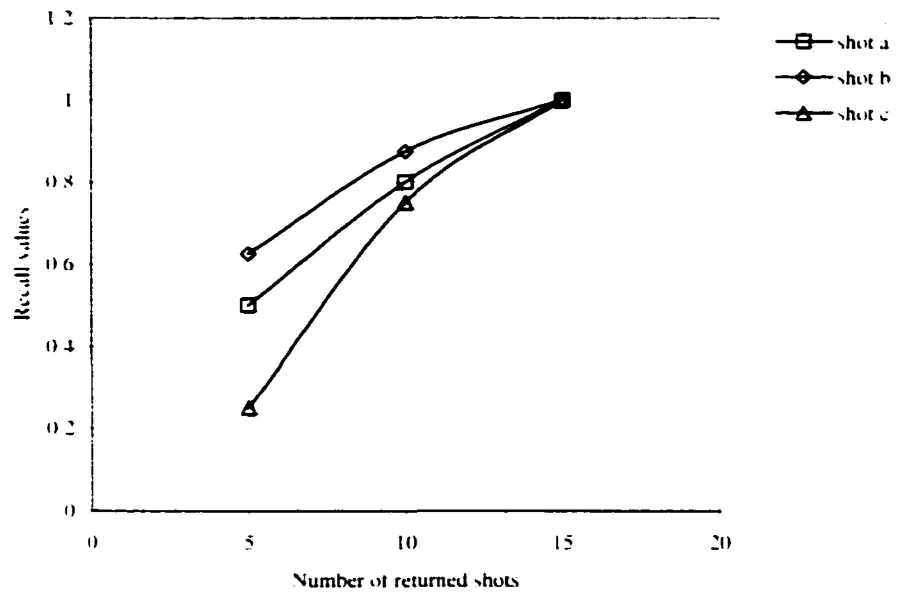

Fig. 42. Recall values for the three unseen query shots.

The recall, precision, and their relation are plotted in Fig. 42, Fig. 43, and Fig. 44 respectively. From Fig. 42 and Fig. 43, it is obvious to remark that the general behavior of the system is kept in which very good values for recall and precision were achieved even in cases where the queries are unseen examples. In particular, Fig. 42 shows that the system can achieve recall values of 1.0 for all the tested queries while the curves in Fig. 43 exhibit acceptable precision trend for the three input queries. In Fig. 44, the relation between average recall and average precision was plotted and we can observe that the degradation in precision is reasonable given that the tested queries are shots that have never been seen before by the system.

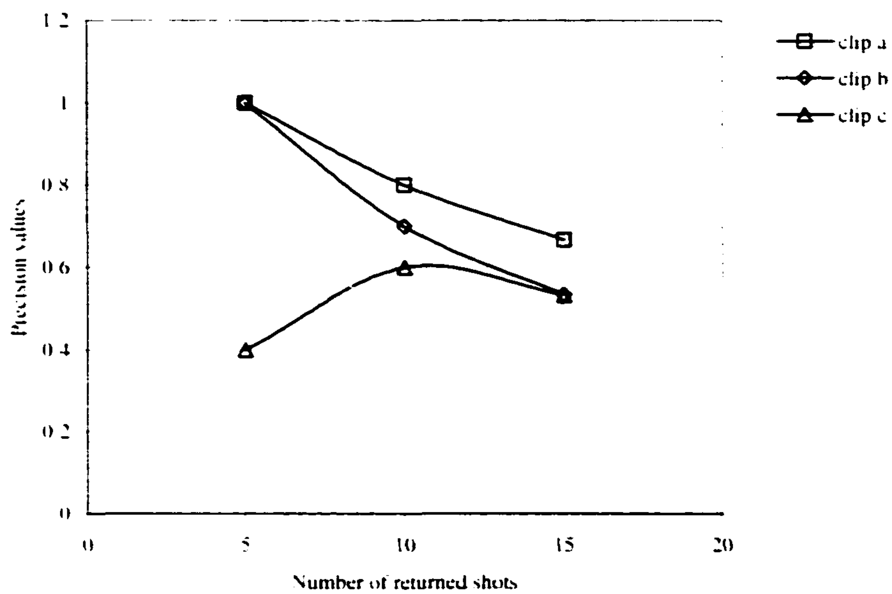

Fig. 43. Precision values for the three unseen query shots.

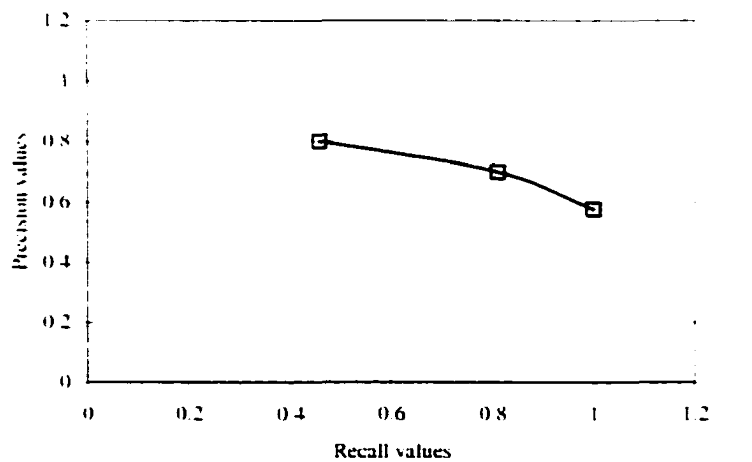

Fig. 44. Recall versus precision for the three unseen query shots.

5.5.3 Evaluating the Performance of the Proposed Similarity Model

In Section 5.5.1 and Section 5.5.2, we study the performance of our system without considering the effect of changing the similarity formula parameters. In those cases, the system selects default values for these parameters to be used in evaluating the similarity using equation (42). In this section, we will analyze and evaluate the usefulness of the proposed similarity model. At first, the system performance will be evaluated using only one similarity factor at a time then the effect of using all of them simultaneously will by studied. To test the visual similarity factor, we first select the ground truth sets then calculate the recall and precision as done before for five shots.

TABLE 51

Recall and Precision as Functions of Returned Shots Number for Shot 2 of soccer

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	14	0.263	1.0
10	10	0	9	0.474	1.0
15	14	1	5	0.737	0.933
20	19	1	0	1.0	0.95

TABLE 52

Recall and Precision as Functions of Returned Shots Number for Shot 1 of crawley

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	12	0.294	1.0
10	10	0	7	0.588	1.0
15	13	2	4	0.765	0.867
20	17	3	0	1.0	0.85

TABLE 53

Recall and Precision as Functions of Returned Shots Number for Shot 3 of adver-sound

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	11	0.313	1.0
10	8	2	8	0.5	0.8
15	12	3	4	0.75	0.8
20	16	4	0	1.0	0.8

TABLE 54

Recall and Precision as Functions of Returned Shots Number for Shot 5 of carton

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	15	0.25	1.0
10	10	0	10	0.5	1.0
15	15	0	5	0.75	1.0
20	20	0	0	1.0	1.0

TABLE 51 through TABLE 55 illustrate the recall and precision values for the five considered shots while TABLE 56 lists their average values. Fig. 45 through Fig. 47 plot these relations where the good performance is remarkable.

TABLE 55

Recall and Precision as Functions of Returned Shots Number for Shot 0 of hoey-v-kill

Number of returned shots allowed	A	B	C	Recall	Precision
5	5	0	11	0.313	1.0
10	9	1	7	0.563	0.9
15	13	2	3	0.813	0.867
20	16	4	0	1.0	0.8

TABLE 56

Average Values of Recall and Precision for the Five Query Shots Considered

Number of returned shots allowed	Recall	Precision
5	0.287	1.0
10	0.525	0.94
15	0.763	0.893
20	1.0	0.88

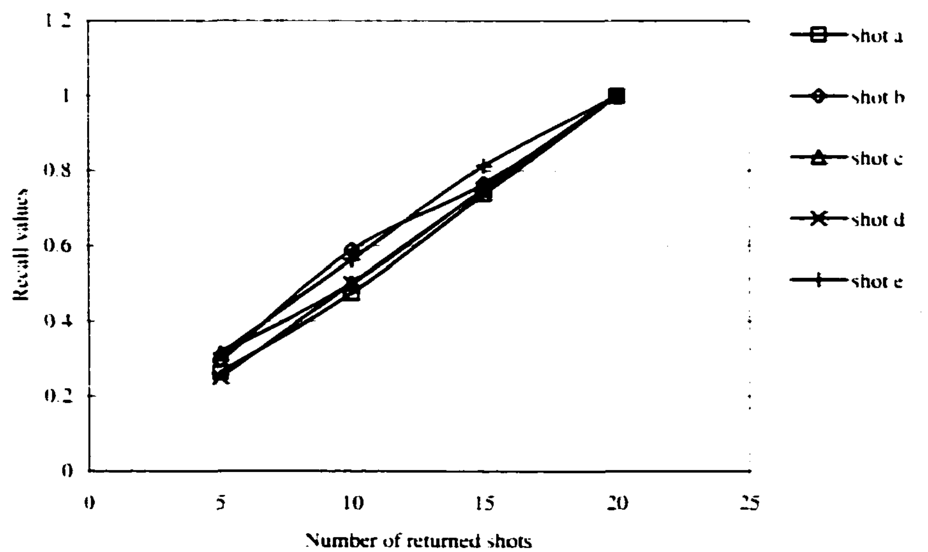

Fig. 45. Recall values for the five query shots considered.

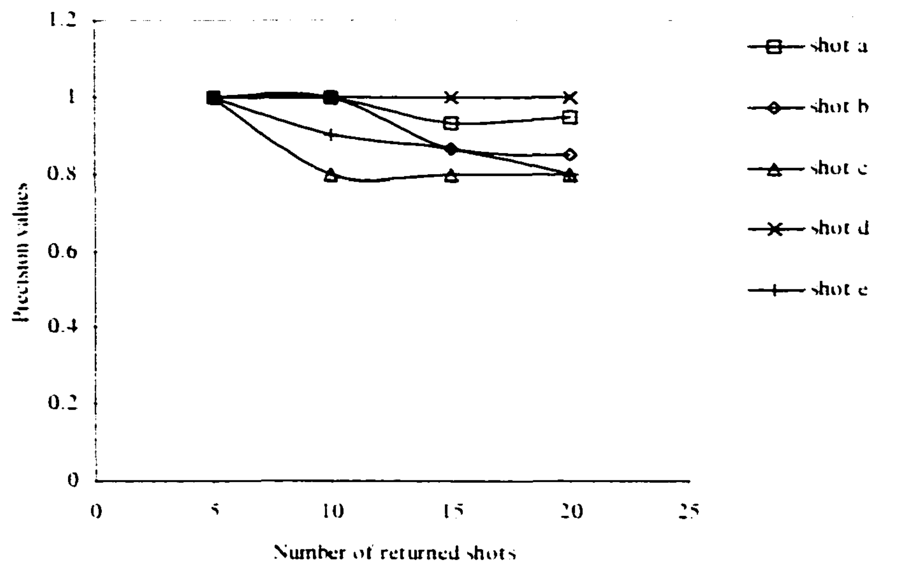

Fig. 46. Precision values for the five query shots considered.

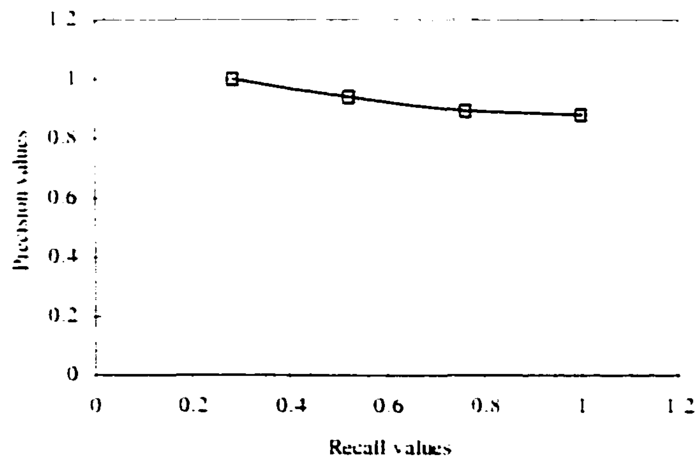

Fig. 47. Recall versus precision for the five query shots considered.

With respect to evaluating the frame rate similarity factor, the task becomes relatively easy for the system because of the straightforward way it can derive this information. Consequently, the obtained recall and precision results were very good. For example, when looking for shots that have the same frame rate as shot 3 of the soccer clip, the system retrieves as hits the 61 shots in the database that has 24 f/s without any misses yielding ideal values for recall and precision. For the duration factor, we obtained very good behavior of the system with ideal values for both recall and precision due to the same reason just mentioned.

In the above set of experiments, the system has shown very good retrieval performance using each of the proposed human-based factors individually. The next question should be "what about its performance when using all of these factors together". As has been mentioned before the system gives the user the flexibility to specify the importance of each of this factor. For sure, the system can not decide in advance what the user wants so it is the responsibility of the user to select the values of these weights. To illustrate the system performance, we use the following examples.

The first shot of the ah6a27 video (a clip from a soccer match) is submitted to the system with the set of similarity weights (W_1, W_2, W_3) chosen as (1.0, 0, 0) that puts all the emphasis on the visual similarity. Fig. 48 shows the results of submitting this query where the first hit is the query shot itself and the second one is a shot from another soccer clip (fai2-3). The third hit is a shot from the racing-boats clip that has a considerable visual similarity (especially in color) with the query shot. By changing the weights to be (0.32, 0.34, 0.34) that gives almost equal weight values to each factor and resubmitting the same query, we got the results shown in Fig. 49. The first hit in Fig. 49 is the same as in Fig. 48 but the second one is the racing-boats shot that was the third hit in Fig. 48. Although the third hit, shot 22 of the documentary clip, seems visually irrelevant to the input query, it has exactly the same frame rate and has almost identical time duration. The involvement of the last two factors (duration and frame rate) directs the system to retrieve not only visually similar shots but also those common to the query in other aspects of similarity. The results of using weight factors of (0.5, 0, 0.5) and (0.3, 0.7, 0) are shown in Fig. 50 and Fig. 51 respectively. The important point to notice here is that the system is flexible and gives the user the ability to express their notion of similarity before submitting the query. This is a good step in trying to measure video similarity on the basis of how humans perceive them. Moreover, this type of human-computer interaction is very effective in improving the performance of any automated retrieval system.

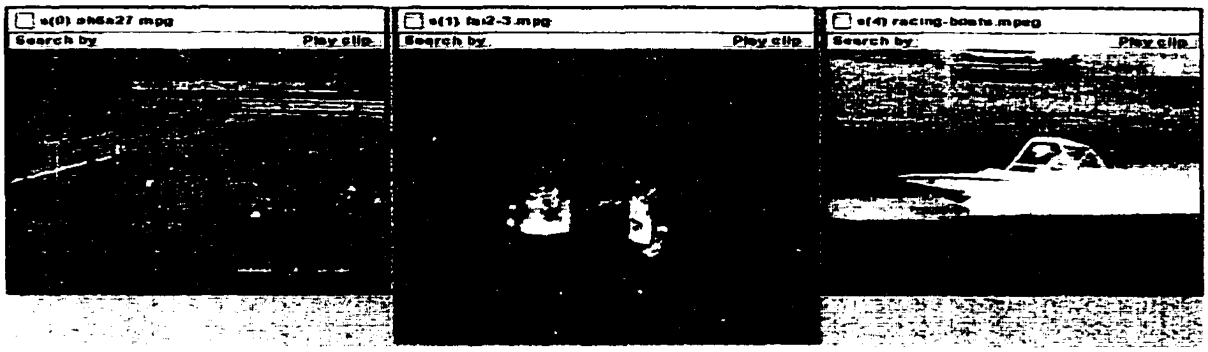

Fig. 48. The first three hits for shot 0 of the ah6a27 clip using weights (1.0, 0, 0).

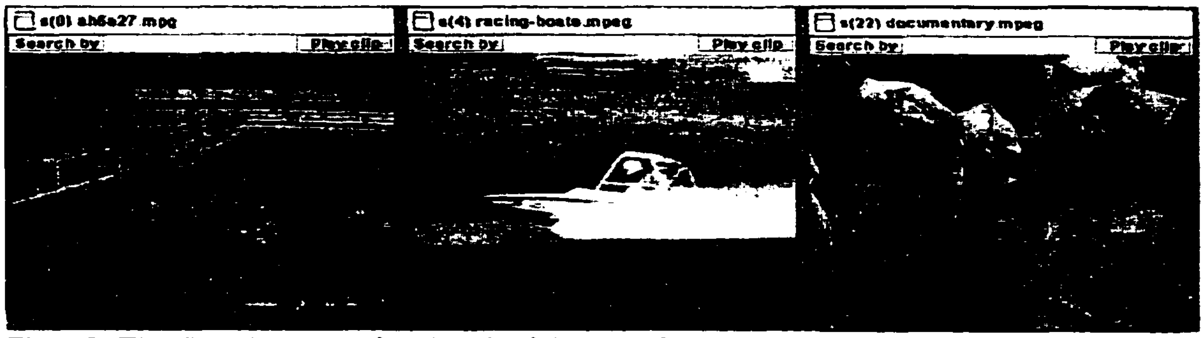

Fig. 49. The first three hits for shot 0 of the ah6a27 clip using weights (0.32, 0.34, 0.34).

Fig. 50. The first three hits for shot 0 of the ah6a27 clip using weights (0.5, 0, 0.5).

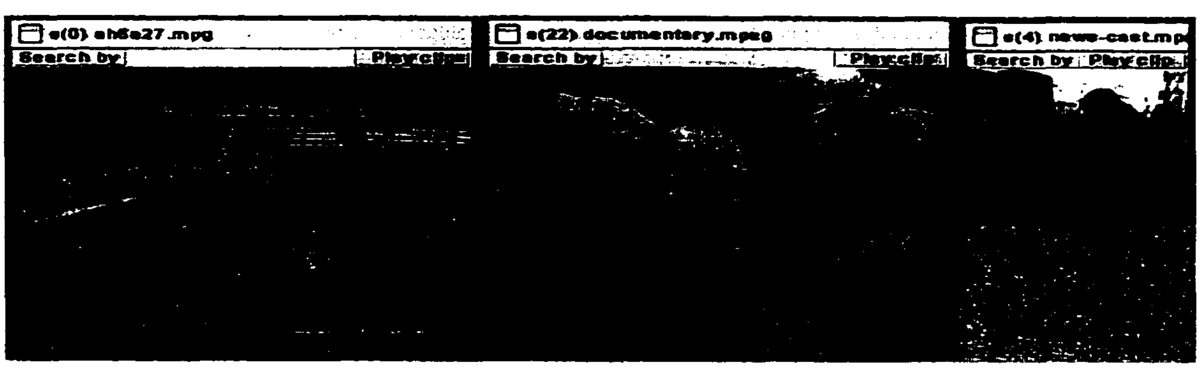

Fig. 51. The first three hits for shot 0 of the ah6a27 clip using weights (0.3, 0.7, 0).

5.5.4 Evaluating the Effectiveness of the Proposed Filtering Scheme

At the end of this performance evaluation section, we need to evaluate the time saving we gained by using the filtering stage to limit the scope of the search space. To accomplish this, a video clip will be submitted to the system as a query while enabling the filtering stage and the search time will be measured. Then, the search time will be measured for the same query but without using the filter and the time saving gained will be calculated. This experiment will be repeated for many input video queries and the average gain will then be determined. To increase the accuracy of our calculation the time we measured is the actual search time while excluding video parsing time and any other overheads. Moreover, all other factors are kept the same such as the machine load, memory and CPU usage, etc. TABLE 57 depicts search times for the 47 clips we considered from our database.

TABLE 57

Saving in Search Times Using the Filtering Stage

Clip name	Search time with filtering (sec)	Search time without filtering (sec)	Saving ratio
carton	4.969	6.701	0.258469
News-cast	11.254	16.364	0.312271
smg-npa-3	17.734	19.126	0.072781
srose-p2	21.852	27.338	0.200673
srose-p4	10.551	12.606	0.163018
tv-show	67.537	87.545	0.228545
adecco	0.64	1.914	0.665622
class	6.162	8.804	0.300091
conf-discussion	16.205	24.531	0.339407
crawle	1.176	3.207	0.633302
dbvath-qcif	1.724	2.433	0.29141
enterprise	0.581	2.537	0.770989
Hoey-v-kill	0.646	2.692	0.76003
ah6a27	1.806	3.666	0.507365
baby-f	0.586	1.228	0.522801
celebration	1.724	2.928	0.411202
Mov0008	0.546	1.315	0.584791
necktw	2.861	3.955	0.276612
Racing-boats	4.57	7.763	0.41131
speed167	3.008	5.177	0.418969
sprint	1.134	3.459	0.67216
tennis1	1.163	1.22	0.046721
Action-movie	2.989	3.774	0.208002
ads	9.581	13.236	0.276141
close	0.469	1.044	0.550766

TABLE 57 Continued

Clip name	Search time with filtering (sec)	Search time without filtering (sec)	Saving ratio
comedy	2.582	3.251	0.205783
corks	4.352	8.003	0.456204
fish	0.598	1.697	0.647613
lions	1.701	2.985	0.430151
sq	0.33	1.556	0.787918
Adver-sound	2.507	2.543	0.014157
ah6a4b	0.344	0.987	0.651469
crawley	0.61	2.015	0.69727
dbvath-cif	0.669	1.882	0.644527
Fai2-3	0.708	1.416	0.5
hoey-v-kill-cif	0.684	2.734	0.749817
soccer	1.316	2.429	0.458213
full_lightbulb	0.336	0.691	0.513748
lightbulb	0.631	0.64	0.014063
peg-in-hole	0.632	0.632	0
reach1	0.345	0.744	0.53629
reach2	0.676	0.771	0.123217
thing	0.302	0.784	0.614796
Thing_gaits	1.327	1.497	0.11356
Thing_learn	0.333	0.725	0.54069
Thing_quest	0.3	0.791	0.620733
two_armed	0.302	0.676	0.553254

From the results in TABLE 57, we can observe that the percentage time saving due to the use of the filtering stage runs from zero in case of peg-in-hole clip to about 79% in clip sq with an average over all the clips we used of 42%. These results represent considerable saving in search time that proves the effectiveness of the filtering approach. Although the potential exclusion of some of the promising clips that may rise due to the use of the filtering stage, the results in TABLE 57 supports the overall advantages of using this scheme. At the same time, any user who cannot tolerate any possible misses can disable this stage on the expense of increasing the system response time.

5.6 Conclusion

In this chapter, we have presented our VCR (Video Content-based Retrieval) system as a step to solve the general problem of accessing video data based on their contents. The design and implementation of the system have been described in details. Moreover, a powerful tool to scope the size of the search space has also been presented. Our main contribution in this stage, the retrieval stage, is the novel similarity model we proposed.

In this model, the similarity of video data is measured based on a number of factors that most probably humans use to judge video similarity. The system also allows users to select the weight associated with each of these factors in order to reflect the need of a particular user in a better way. The performance of the filtering stage has been evaluated and a considerable speed up gain of about 42% on average has been achieved when using it. Evaluating the retrieval stage performance was achieved through measuring the recall and precision of the returned results in which very good values for recall and precision were obtained whether the query is part of the database or not. In addition, our evaluation of the performance of the proposed similarity model has yielded very good retrieval rate and accuracy. One important issue we need to highlight here is that the evaluation method we used not only measures the performance of the retrieval stage but also evaluates the performance of all the stages of the developed system. These stages include shot boundary detection scheme, key frames selection algorithms, and the indexing techniques. Thus, the results in this chapter represent a sound proof of the overall success and usefulness of the proposed content-based video retrieval system we introduced in this thesis.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

Providing effective video content-based retrieval systems is gaining the attention of the research community. In this thesis. we presented a novel system to achieve that task and avoid the shortcomings of other approaches. Our research opens new horizons of looking at the problem of retrieving video data based on their contents and reveals new questions to be addressed. In this chapter. we summarize some relevant conclusions and discuss future extensions to this work.

6.1 Conclusions

The current widespread use of multimedia applications is the result of the dramatic increase in the available Internet bandwidth and machines performances. These applications are continuously generating tremendous amounts of multimedia data that need to be properly organized and stored to allow effective access to them. The effectiveness of the search system is of crucial importance to the overall performance of the retrieval process. It is the major factor in determining the success or failure in obtaining the required information. Searching video databases using the conventional keyword-based search technique is completely inadequate due to the unstructured nature of these data. Likewise. current techniques for content-based access of video data lack one or more of the desirable characteristics needed to come up with an effective content-based system for retrieving video data. These characteristics include reliability. efficiency. extensibility. scalability. and accuracy.

Realizing the significance of the effectiveness of the search technique. we have presented. in this thesis. a new paradigm capable of solving the problem of content-based retrieval of video data while avoiding the main drawbacks of other techniques. It addresses four subsystems that constitute a fully content-based video retrieval system. the shot boundary detection stage. the key frames selection module. the indexing stage. and the retrieval subsystem. Here are our concluding remarks regarding these subsystems:

- **Shot Boundary Detection:** To achieve efficient manipulation of video data, we started by abstracting the original video stream through the use of its DC sequence. Almost all further processing stages are working upon that sequence. After that, we introduced a novel approach that uses a feedforward neural network module to accomplish the task of shot boundary detection. The network learns the mapping information from a set of training samples during an off-line training phase using the momentum back-propagation as a learning algorithm. It is then used during its recall phase to identify shot boundaries in video streams. The shot boundary detection stage achieved its reliability through the novel use of the supervised learning model to solve the problem at hand. Moreover, outstanding efficiency is primarily achieved as a result of using the instantaneous recall phase of the neural network and working upon an abstract version of the video data. This stage's extensibility is another desirable feature that can be easily exploited by retraining the network using new representative samples of the video types we are concerned with. Experimental results showed the superior efficiency and reliability of the developed shot boundary detection stage compared to other techniques that perform the same task. Our approach detection speed was more than 8 times faster than other techniques while the detection percent achieved was 97% with 2.6% false alarms.
- **Key Frames Selection:** The shot boundary detection stage organizes a video stream into a set of meaningful and manageable units. Still, each of these shots has a tremendous amount of data that need to be summarized by an efficient and effective technique. In order to accomplish this summarization task, two adaptive algorithms (AFS and ALD) were developed to solve the problem of key frames selection. The algorithms are simple; thus, providing high reliability and efficiency. Moreover, to enhance the performance of the proposed algorithms, both employed two-level adaptation strategy to adjust the selection decision based on varying conditions. The first level handles the problem of different video dimensions. While, in the second adaptation level, each algorithm measures the amount of activity in each shot and either increases or decreases the sampling frequency in accordance with the shot activity level. This way, they efficiently selected the minimal expressive sets of key frames that are capable of capturing the salient characteristics in the video stream. For

instance, the AFS algorithm managed to select a near optimal set of key frames to represent a 6139-frame clip in two seconds while running on a Sparc ultra 60 machine. A comparison between the performances of the developed algorithms was also given yielding better performance for the AFS over the ALD algorithm.

- **Indexing stage:** To construct the metadata describing original video streams, two low-level features, color and texture, were used to derive content indexes from the selected key frames. The color feature of each key frames was represented via the use of that frame's color histogram which was derived after converting the DC frame from the YCbCr to the RGB color space. The color feature extraction algorithm effectively reduces the dimension of the generated histograms and applies a normalized histogram intersection policy to measure frames similarity. With respect to the texture feature, it has been calculated using the Gabor wavelet transform where the mean and standard deviation for each scale and orientation were used as elements of the texture feature vector. A normalized distance measure was then used to calculate texture similarity. The color histogram and the texture feature vector are the basic components of the metadata representing each key frame. These metadata were used instead of the original video data in any further processing or similarity matching operations.
- **Retrieval Stage:** Improving the effectiveness at which the required information is retrieved is the basic objective of any video content-based retrieval system. This effectiveness can be maximized if the similarity model can perform its matching operations in a similar way performed by human beings. The developed retrieval stage considered this vital issue and proposed the use of a number of human-based factors in determining the similarity of video data. These factors include video content visual similarity, shot order, frame rate, and shot duration. This novel approach captured human notion in measuring multimedia data similarity; thus, improving the performance of the whole system. The retrieval stage also provided the users with full control in directing the search according to what they are interested in. In addition, a two-level search procedure was provided to enhance response time and scalability. It achieved a speed up gain of about 42% on average. The performance of the whole system has been tested experimentally and we achieved very good retrieval

rate and accuracy that reached its ideal value (1.0) in some cases. Moreover, the retrieval model has proven its usefulness in allowing users to express their own needs and preferences.

The developed system has been implemented from scratch in Java and tested on both Windows 2000 and the Solaris operating systems. During our experimentations we evaluated a number of dominant parameters of the algorithms in each stage. For instance, both the efficiency and the reliability of the shot boundary detection stage have been evaluated. Moreover, the achieved results were compared with other systems' results, when available, showing the superiority of our proposed techniques. In a nutshell, the developed system manifests, through extensive experimentations, its reliability, efficiency, extensibility, interoperability, effectiveness, and accuracy in retrieving the required video information from the video library.

6.2 Future Work

The proposed techniques are one step towards improving the performance of video content-based retrieval systems and, as many research efforts, the doors are opened for further improvements and extensions. In this section, we explain some possible extensions to each stage of the developed system and discuss briefly some new features that can be added to the current architecture. Finally, we comment on the correlation between our system and the new emerging MPEG-7 [11], [41], [84] video standard.

- The developed shot boundary detection paradigm was designed to detect the prevailing type of shot boundaries, cuts. Currently, there is an increasing trend in video production that uses gradual transition effects such as dissolve, fad in, fade out, and wipes. More work is needed to introduce a new algorithm or extend the proposed one for our system to parse video streams containing these transition effects. Further efforts are also required to investigate the performance of the proposed shot boundary detection module when applying to very-large video databases. One parameter that can improve the performance in such situations is the proper choice of the training set for the neural network.
- Although the developed key frames selection algorithms are both accurate and efficient, there is a room for extending and improving the performance of this stage.

One possible direction is the development of an efficient algorithm that detects the semantic objects inside video frames. By detecting semantic objects. the algorithm can determine which key frames are the best to be selected in order to represent the shot salient features and at the same time avoid the choice of redundant frames. More work is needed to comprehensively measure the accuracy of the developed key frames selection algorithms for other video types not tested during our experimentations.

- More than one improvement is foreseen for the proposed indexing stage. The performance of the texture extraction algorithm. during our experimentations. precludes its use in on-line query processing. Therefore. it is quite useful to come up with an efficient algorithm for texture extraction that can perform its task in an acceptable time period. Other low-level features can also be used in order to widen the applicability of the proposed retrieval system. For example. shape and spatial relationships among objects are two candidate features that can be used to extend our metadata representation of key frames. Exploring the temporal dimension of video streams is another direction in improving the performance of the indexing system. For instance. camera work can be detected and used as another component of the metadata. Finally. multi-dimension indexing structure is one of the active areas of research [9] that can be further explored.
- A possible enhancement to the retrieval subsystem is the introduction of a relevance feedback algorithm where the user can give scores to the returned search results. These scores will be used to guide the next search phase with the ultimate goal of producing more relevant results. In the proposed similarity matching model. we tried our best to come up with a simple but efficient model that describes the way humans perceive video data similarity. More work can be directed at exploring the usefulness of other human-based similarity matching factors and proposing appropriate methods for integrating them into the developed system.
- On the other hand. new components can be added to our system to widen its applicability and improve its usability. One example is the addition of a new browsing technique that uses three-dimension representation of the search results in a trial to improve the user visualization. Another component that can be added to the

system is a new DC extraction module that can parse another video input format: for instance. Quick time video. Intel Indeo format. or Microsoft video for Windows in addition to the current DC extraction module. One advantage of our system is its object-oriented architecture that allows for easy extension by just plugging a new component into its flexible structure.

- One last comment is related to how the proposed system can handle video streams encoded using the current emerging multimedia content description interface. the MPEG-7. Our system has a unified objective with the MPEG-7 standard in which both aim to provide effective search and retrieval techniques of stored multimedia data. Obviously. the new tools provided by MPEG-7 will make the video analysis task easier than in the case of MPEG-1 (used by our system). Meanwhile. an important observation is that MPEG-7 does not specify or standardize how the features are extracted or applied. This property allows our proposed techniques to be applicable to MPEG-7 streams provided that the DC sequence extraction module is modified to parse MPEG-7 data.

REFERENCES

[1] M. Abbadi. *Content-Based Indexing and Searching in Image Databases*. Ph.D. thesis. Dept. of Computer Science. George Washington University. 2000.

[2] M. Abdel-Mottaleb. N. Dimitrova. R. Desai. and J. Martino. "CONIVAS: Content-based Image and Video Access System." *Proc. ACM int'l Conf. Multimedia*. pp. 427-428. Nov. 1996.

[3] G. Ahanger and T. Little. "A Survey of Technologies for Parsing and Indexing Digital Video." *J. Visual Communication and Image Representation*. vol. 7. no. 1. pp. 28-43. March 1996.

[4] J. Assflag and P. Pala. "Querying by Photographs: A VR Metaphor for Image Retrieval." *IEEE Multimedia*. vol. 7. no. 1. pp. 52-59. March 2000.

[5] Y. Avrithis. A. Doulamis. N. Doulamis. and S. Kollias. "A Stochastic Framework for Optimal Key Frame Extraction from MPEG Video Databases." *J. Computer Vision and Image Understanding*. vol. 75. no. 1. pp. 3-24. 1999.

[6] R. Baldwin. "Portals on Edge." *IEEE Multimedia*. vol. 7. no. 1. pp. 10-11. March 2000.

[7] R. Beale and T. Jackson. *Neural Computing: An Introduction*. New York: IOP Publishing Ltd.. 1991.

[8] Berkeley Digital Library Project. http://elib.cs.berkeley.edu/admin/proposal/proj-des/proj-des.html.

[9] A. Bimbo. *Visual Information Retrieval*. Morgan Kaufmann Publishers. Inc.. San Francisco. 1999.

[10] R. Brunelli. O. Mich. and C. Modena. "A Survey on the Automatic Indexing of Video Data." *J. Visual Communication and Image Representation*. vol. 10. no. 2. pp. 78-112. June 1999.

[11] S-F. Chang. T. Sikora. and A. Puri. "Overview of the MPEG-7 Standard." *IEEE Trans. Circuits and Systems for Video Technology*. vol. 11. no. 6. pp.688-695. 2001.

[12] S-F. Chang. W. Chen. H. Meng. H. Sundaram. and D. Zhong. "A Fully Automated Content Based Video Search Engine Supporting Spatio-Temporal Queries." *IEEE Trans. Circuits and Systems for Video Technology*. vol. 8. no. 5. pp. 602-615. Sept. 1998.

[13] S-F. Chang. "Compressed Domain Techniques for Image/Video Indexing and Manipulation." *Proc. IEEE int'l Conf. Image Processing*. vol. 1. pp. 314-317. Oct. 1995.

[14] J. Chen. C. Taskiran. A. Albiol. E. Delp. and C. Bouman. "ViBE: A Video Indexing and Browsing Environment." *Proc. SPIE/IS&T Conf. Multimedia Storage and Archiving Systems IV*. vol. 3846. pp. 148-164. Sept. 1999.

[15] S. Cheung and A. Zakhor. "Efficient Video Similarity Measurement and Search." *Proc. IEEE Int'l Conf. Image Processing*. pp. 85-89. 2000.

[16] S. Cheung and A. Zakhor. "Video Similarity Detection with Video Signature Clustering." *Proc. IEEE Int'l Conf. Image Processing*. pp. 649-652. 2001.

[17] M. Christel. A. Olligschlaeger. and C. Huang. "Interactive Maps for Digital Video Library." *IEEE Multimedia*. vol. 7. no. 1. pp. 60-67. March 2000.

[18] M. Christel. T. Kanade. M. Mauldin. R. Reddy. M. Sirbu. S. Stevens. and H. Wactlar. "Informedia Digital Video Library." *Comm. of the ACM*. vol. 38. no. 4. pp.57-58. April 1995.

[19] L. Davis. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold. 1991.

[20] M. Davis. "Media Streams: An Iconic Visual Language for Video Annotation." *Proc. IEEE Workshop on Visual Languages*. Norway. pp. 196-201. August 1993.

[21] E. Delp. "Video and Image Databases: Who Cares?." *Proc. SPIE/IS&T Conf. Storage and Retrieval for Image and Video Databases VII*. vol. 3656. San Jose. CA. pp. 274-277. Jan. 1999.

[22] Y. Deng and B. Manjunath. "Content-based Search of Video Using Color. Texture. and Motion." *Proc. IEEE int'l Conf. Image Processing*. Washington. DC. vol. 2. pp. 534-537. 1997.

[23] M. Drew and J. Au. "Video Key Frame Production by Efficient Clustering of Compressed Chromaticity Signatures." *Proc. ACM int'l Conf. Multimedia*. pp. 365-367. 2000.

[24] A. Drozdek. *Elements of Data Compression*. Brooks/Cole Publishing. 2002.

[25] Waleed Farag and Hussein Abdel-Wahab. "A New Paradigm for Analysis of MPEG Compressed Videos." *J. Network and Computer Applications*. vol. 25. no. 2. pp. 109-127. 2002.

[26] Waleed Farag and Hussein Abdel-Wahab. "A New Paradigm for Detecting Scene Changes on MPEG Compressed Videos." *Proc. IEEE Int'l Symposium Signal Processing and Information Technology*. Cairo. Egypt. pp. 153-158. Dec. 2001.

[27] Waleed Farag and Hussein Abdel-Wahab. "Adaptive Key Frames Selection Algorithms for Summarizing Video Data." *Proc. 6th Joint Conference on Information Sciences*. Durham. NC. pp. 1017-1020. March 2002.

[28] Waleed Farag and Hussein Abdel-Wahab. "Techniques for Indexing MPEG Compressed Videos." *Technical Report TR_2002_01. Dept. of Computer Science. Old Dominion Univ.*. March 2002.

[29] Waleed Farag and Hussein Abdel-Wahab. "Video Content-based Retrieval Techniques." a book chapter to appear in. *Multimedia Systems and Content-based Image Retrieval*. Idea Group Inc.

[30] Waleed Farag, I. Ziedan, M. syiam, and M. Mahmoud, "Architecture of Neural Networks Using Genetic Algorithms," *Proc. 5th Int'l Conf. Artificial Intelligence Applications (5th ICAIA)*, Cairo, Egypt, March 1997.

[31] M. Flinkner, et. al., "Query by Image and Video Content: the QBIC System," *IEEE Computer*, vol. 28, no. 9, pp. 23-32, Sept. 1995.

[32] A. Girgensohn and J. Boreczky, "Time-constrained Key Frame Selection Technique," *Proc. IEEE int'l. Conf. Multimedia Computing and Systems*, pp. 756-761, 1999.

[33] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989.

[34] A. Gupta and R. Jain, "Visual Information Retrieval," *Comm. of ACM*, vol. 40, no. 5, pp. 70-79, 1997.

[35] A. Hampapur, R. Jain, and T. Weymouth, "Digital Video Segmentation," *Proc. ACM Int'l Conf. Multimedia*, pp. 357-364, Oct. 1994.

[36] A. Hanjalic and H-J Zhang, "Optimal Shot Boundary Detection Based on Robust Statistical Models," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems*, pp. 710-714, 1999.

[37] W. Hsu, S. Chua, and H. Pung, "An Integrated Color-spatial Approach to Content-based Image Retrieval," *Proc. ACM Int'l Conf. Multimedia*, pp. 305-313, 1995.

[38] F. Idris and S. Panchanathan, "Review of Image and Video Indexing Techniques," *J. Visual Communication and Image Representation*, vol. 8, no. 2, pp. 146-166, June 1997.

[39] M. Irani, D. Anandan, and S. Hsu, "Mosiac Based Representation of Video Sequences and their Applications," *Proc. IEEE 5th int'l Conf. Computer Vision*, pp. 605.611, 1995.

[40] ISO/IEC 11172-2:1993/Cor 2:1999: Information Technology – "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s", Part 2: Video.

[41] R. Johnson, "Multimedia Databases and MPEG-7," *J. Electronics and Communication Engineering*, vol. 13, no. 3, pp. 98-99, 2001.

[42] M. Kankanhalli and T-S. Chua, "Video Modeling Using Strata-based Annotation," *IEEE Multimedia*, vol. 7, no. 1, pp. 68-74, March 2000.

[43] N. Kosugi, *et al.*, "Content-based Retrieval Applications on a Common Database Management System," *Proc. ACM Int'l Conf. Multimedia*, pp. 599-600, 2001.

[44] S. Krishnamachari and M. Adbel-Mottaleb, "Image Browsing Using Hierarchical Clustering," *Proc. 4th IEEE Symposium Computers and Communications*, vol. 4, pp. 301-309, July 1999.

[45] T. Kuo and A. Chen, "Content-based Query Processing for Video Databases," *IEEE Trans. Multimedia*, vol. 2, no. 1, pp. 1-13, March 2000.

[46] S-W. Lee, Y-M. Kim, and S. Choi, "Fast Scene Change Detection Using Direct Feature Extraction from MPEG Compressed Videos," *IEEE Trans. Multimedia*, vol.2, no. 4, pp. 240-254, 2000.

[47] D. LeGall, "MPEG: a Video Compression Standard for Multimedia Applications" *Comm. of ACM*, vol. 34, no. 4, pp. 46 – 58, May 1991.

[48] M. Lew, *Principles of Visual Information Retrieval*, edited, Springer-Verlag, 2001.

[49] X. Liu, Y. Zhuang, and Y. Pan, "A New Approach to Retrieve Video by Example Video Clip," *Proc. ACM int'l Conf. Multimedia*, pp. 41-44, 1999.

[50] C. Low, Q. Tian, and H-J Zhang, "An Automatic News Video Parsing, Indexing, and Browsing System," *Proc. ACM int'l Conf. Multimedia*, pp. 425-426, 1996.

[51] H. Luo and A. Eleftheriadis, "Designing an Interactive Tool for Video Object Segmentation and Annotation: Demo Abstract," *Proc. ACM Int'l Conf. Multimedia*, p. 196, 1999.

[52] W. Ma, and B. Manjunath, "NeTra: A Tool for Navigating Large Image Databases," *Proc. IEEE int'l Conf. on Image Processing*, vol. 1, pp. 568-572, 1997.

[53] B. Manjunath and W. Ma, "Texture Features for Browsing and Retrieval of Image Data," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837-842, Aug. 1996.

[54] J. Meng, Y. Juan, and S-F. Chang, "Scene Change Detection in an MPEG Compressed Video Sequence," *Proc. IS&T/SPIE Symposium on Digital Video Compression: Algorithms and Technologies*, vol. 2419, pp. 14-25, Feb. 1995.

[55] T. Minka, "An Image Database Browser that Learns from User Interaction," Master's thesis, MIT Media Laboratory, 1996.

[56] T. Minka, and R. Picard, "Interactive Learning Using a Society of Models," *J. Pattern Recognition*, vol. 30, no. 4, pp. 565-582, 1997.

[57] J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall, *MPEG Video: Compression Standard*, Chapman and Hall, 1997.

[58] A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-video Search for Object Appearance," *Proc. Visual Database Systems 2*, pp. 113-127,1991.

[59] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape," *Proc. SPIE Storage and Retrieval for Image and Video Databases*, vol. 1908, pp. 173-187, Feb. 1993.

[60] W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.

[61] A. Petland, R. Picard, and S. Sclaroff, "Photobook: Tools for Content-based Manipulation of Image Databases," *Proc. SPIE: Storage and Retrieval for Image and Video Databases II*, vol. 2185, pp. 34-47, Feb. 1994.

[62] R. Picard and T. Minka. "Vision Texture for Annotation." *J. Multimedia Systems*. vol. 3. pp. 3-14. 1995.

[63] R. Picard. T. Minka. and M. Szummer. "Modeling User Subjectivity in Image Libraries." *Proc. IEEE int'l Conf. Image Processing*. vol. 2. pp. 777-780. 1996.

[64] Query By Image Content system. **http://wwwqbic.almaden.ibm.com**.

[65] Y. Rui. T. Huang. and S. Mehrotra. "Browsing and Retrieving Video Content in a Unified Framework." *Proc. IEEE Workshop on Multimedia Signal Processing*. pp. 9-14. Dec. 1998.

[66] Y. Rui. T. Huang. and S. Mehrotra. "Relevance Feedback Techniques in Interactive Content-based Image Retrieval." *Proc. SPIE Conf. Image Science and Technology (IS&T)*. pp. 25-36. 1998.

[67] Y. Rui. T. Huang. and S-F. Chang. "Image Retrieval: Current Techniques. Promising Directions. and Open Issues." *J. Visual Communication and Image Representation*. vol. 10. no. 1. pp. 39-62. March 1999.

[68] E. Rumelhart. G. E. Hinton. and R. J. Williams. "Learning Internal Representation by Error Propagation." *In Parallel Distributed Processing*. D. E. Rumelhart. et al.. ed. Cambridge. MA: MIT Press. pp. 318-362. 1986.

[69] E. Sahouria and A. Zakhor. "Motion Indexing of Video." *Proc. IEEE int'l Conf. Image Processing*. Washington. DC. vol. 2. pp. 526-529. 1997.

[70] S. Santini and R. Jain. "Similarity Measures." *IEEE Trans. Pattern Analysis and Machine Intelligence*. vol. 21. no. 9. pp. 871-883. Sept. 1999.

[71] K. Shen and E. Delp. "A Fast Algorithm for Video Parsing Using MPEG Compressed Sequences." *Proc. IEEE int'l Conf. Image Processing*. vol. 2. pp. 252-255. 1995.

[72] M. Sheneier and M. Abdel-Mottaleb. "Exploiting the JPEG Compression Scheme for Image Retrieval." *IEEE Trans. Pattern Analysis and Machine Intelligence*. vol. 18. no. 8. pp. 849-853. Aug. 1996.

[73] A. Singh. *Optical Flow Computation: A Unified Perspective*. IEEE Computer Society Press. 1991.

[74] J. Smith and S-F. Chang. "VisualSEEK: A Fully Automated Content-based Image Query System." *Proc. ACM int'l Conf. Multimedia*. pp. 87-98. 1996.

[75] S. Smoliar and H-J. Zhang. "Content-based Video Indexing and Retrieval." *IEEE Multimedia*. vol. 1. no. 2. pp.62-72. Summer 1994.

[76] Stanford Digital Library Group. "The Stanford Digital Library Project." *Comm. of ACM*. vol. 38. no. 4. pp. 59-60. April 1995.

[77] Sun Microsystems. Java language. http://java.sun.com/products.

[78] M. Swain and D. Ballard. "Color Indexing." *J. Computer Vision*. vol. 7. no. 1. pp. 11-32. 1991.

[79] Y. Tan, S. Kulkarni, and P. Ramadge, "A Framework for Measuring Video Similarity and its Application to Video Query by Example," *Proc. IEEE Int'l Conf. Image Processing*, pp. 106-110, 1999.

[80] Y. Tonomura, "Video Handling Based on Structured Information for Hypermedia Systems," *Proc. ACM int'l Conf. Multimedia Information Systems*, pp. 333-344, 1991.

[81] Y. Tonomura, A. Akutsu, K. Otsuji, and T. Sadakata, "VideoMap and VideoSpaceIcon: Tools for Anatomizing Video Content," *Proc. ACM INTERCHI*, pp. 131-141, 1993.

[82] S. Uchihashi, J. Foote, A. Girgensohn, and J. Boreczky, "Video Manga: Generating Semantically Meaningful Video Summaries," *Proc. ACM Int'l Conf. Multimedia*, pp. 383-392, 1999.

[83] N. Vasconcelos and A. Lippman, "Towards Semantically Meaningful Spaces for the Characterization of Video Content," *Proc. IEEE int'l Conf. Image Processing*, vol. II, pp. 542-545, Oct. 1997.

[84] V. Vinod and A. Lindsay, "MPEG-7: Its Impact on Research, Industry, and the Consumer," *Proc. IEEE int'l Conf. Multimedia Computing and Systems*, vol. 2, pp. 406-407, 1999.

[85] Virage Video Indexing System, **http://www.virage.com**.

[86] J. Wachman, "A Video Browser that Learns by Example," Master's thesis, MIT Media Laboratory, 1997.

[87] H. Wactlar, M. Christel, Y. Gong, and A. Hauptmann, "Lessons Learned from Building a Terabyte Digital Video Library," *IEEE Computer*, vol. 32, no. 2, pp. 66-73, Feb. 1999.

[88] G. Wallace, "The JPEG Still Picture Compression Standard," *Comm. of ACM*, vol. 34, no. 4, pp. 30 – 44, 1991.

[89] L. Wilcox, *et al.*, "MBase: Indexing, Browsing, and Playback of Media at FXPAL," *Proc. ACM Int'l Conf. Multimedia*, p 204, 1999.

[90] W. Wolf, "Key Frame Selection by Motion Analysis," *Proc. IEEE int'l. Conf. Acoustic, Speech, and Signal Processing*, vol. 2, pp. 1228-1231, 1996.

[91] Y. Wu, Y. Zhuang, and Y. Pan, "Content-based Video Similarity Model," *Proc. ACM int'l Conf. Multimedia*, pp. 465-467, 2000.

[92] P. Yamuna and K. Candan, "Similarity-based Retrieval of Temporal Documents," *Proc. ACM Int'l Conf. Multimedia*, pp. 243-246, 2000.

[93] B-L. Yeo and B. Liu, "On the Extraction of DC Sequence from MPEG Compressed Video," *Proc. IEEE int'l Conf. Image Processing*, vol. 2, pp. 260-263, 1995.

[94] B-L. Yeo and B. Liu, "Rapid Scene Analysis on Compressed Video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533-544, Dec. 1995.

[95] M. Yeung and B. Liu. "Efficient Matching and Clustering of Video Shots." *Proc. IEEE int'l Conf. Image Processing*. Washington. DC. vol. 1. pp. 338-341. Oct. 1995.

[96] A. Yoshitaka and T. Ichikawa. "A Survey on Content-based Retrieval for Multimedia Databases." *IEEE Trans. Knowledge and Data Eng.*. vol. 11. no. 1. pp. 81-93. Jan. 1999.

[97] H-J. Zhang. J. Wu. D. Zhong. and S. Smoliar. "An Integrated System for Content-based Video Retrieval and Browsing." *J. Pattern Recognition*. vol. 30. no. 4. pp. 643-658. 1997.

[98] H-J. Zhang. A. Kankanhalli. S. Smoliar. and S. Tan. "Automatically Partitioning of Full-motion Video." *J. Multimedia Systems*. vol. 1. no. 1. pp. 10-28. 1993.

[99] H-J. Zhang. C. Low. Y. Gong. S. Smoliar. and S. Tan. "Video Parsing Using Compressed Data." *Proc. SPIE Image and Video Processing II*. vol. 2182. pp. 142-149. 1994.

[100] H-J. Zhang. J. Wang. and Y. Altunbasak. "Content-based Video Retrieval and Compression: A Unified Solution." *Proc. IEEE int'l Conf. Image Processing*. Washington. DC. vol. 1. pp. 13-16. 1997.

[101] H-J. Zhang. S. Tan. S. Smoliar. and Y. Gong. "Automatic Parsing and Indexing of News Video." *J. Multimedia Systems*. vol. 2. pp.256-266. 1995.

[102] D. Zhong and S-F. Chang. "Spatio-temporal Video Search Using the Object Based Video Representation." *Proc. IEEE int'l Conf. Image Processing*. Washington. DC. vol. 1. pp. 21-24. 1997.

[103] X. Zhou and T. Huang. "Relevance Feedback in Content-based Image Retrieval: Some Recent Advances." *Proc. 6th Joint Conf. On Information Sciences*. pp. 15-18. 2002.

[104] Y. Zhuang. Y. Rui. T. Huang. and S. Mehrotra. "Adaptive Key Frame Extraction Using Unsupervised Clustering." *Proc. IEEE int'l Conf. Image Processing*. Chicago. IL. pp. 866-870. Oct. 1998.

[105] J. Zurada. *Introduction to Artificial Neural Systems*. West Publishing Company. 1992.

APPENDIX A

KEY FRAMES SELECTION RESULTS FOR LONGER CLIPS

A.1 Additional Results Using the Algorithm in Section 3.2.1

Below are the results obtained by applying the algorithm in Section 3.2.1 to two relatively large clips from TABLE 3 called ads and conf-discussion.

TABLE A.1

KFs Selection for the ads Clip Using δ_f =150.000 without any Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-65	66	2	0, 37
1	66-97	32	1	66
2	98-258	161	2	98, 176
3	259-304	46	1	259
4	305-407	103	2	305, 369
5	408-437	30	1	408
6	438-579	142	3	438, 517, 574
7	580-614	35	2	580, 611
8	615-650	36	2	615, 646
9	651-674	24	1	651
10	675-758	84	2	675, 721

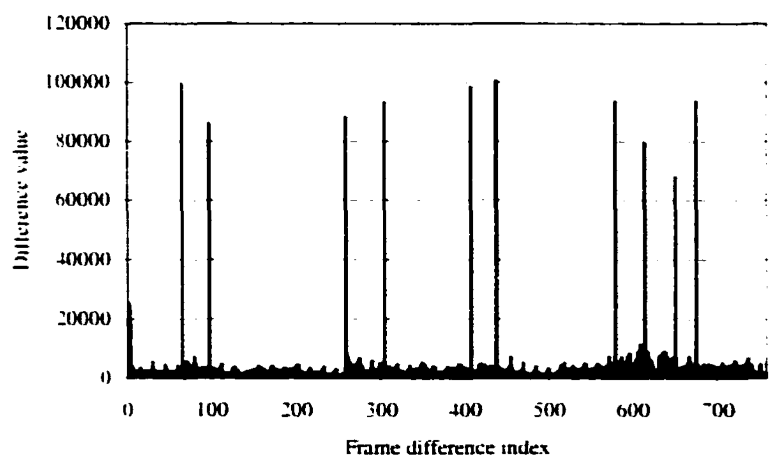

Fig. A.1. The frame difference graph for the ads video.

In TABLE A.1. there is a large variation in the illumination of the first three frames (due to a gradual transition effect) that affects the choice of two KFs for the first shot although it is a still one. The illumination change can be noticed in the frame difference diagram of that clip shown in Fig. A.1 as two relatively large peaks at the start of the first shot. The second shot is a low activity and short one so one KF is appropriate. For other shots such as shots 4 and 6. the algorithm chooses more than the necessary numbers of KFs due to their lengths.

TABLE A.2 reports the KFs selection results for the conf-discussion video clip whose frame difference graph is shown in Fig A.2. We can remark that many of the needed key frames have not been selected in that table. This is due to the small size of this clip and the same comments made about TABLE 15 can also apply here.

TABLE A.2

KFs Selection for the conf-discussion Clip Using δ_l =150.000 without any Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-48	49	1	0
1	49-456	408	2	49. 232
2	457-635	179	1	457
3	636-688	53	1	636
4	689-1139	451	2	689. 948
5	1140-1213	74	1	1140
6	1214-1279	66	1	1214
7	1280-1542	263	2	1280. 1531
8	1543-1593	51	1	1543
9	1594-1713	120	1	1594
10	1714-1959	246	1	1714
11	1960-2262	303	1	1960
12	2263-2473	211	1	2263
13	2474-2595	122	1	2474
14	2596-3208	613	2	2596. 3015
15	3209-3255	47	1	3209
16	3256-4202	947	3	3256. 3640. 4038
17	4203-4310	108	1	4203
18	4311-4500	190	1	4311
19	4501-4782	282	1	4501

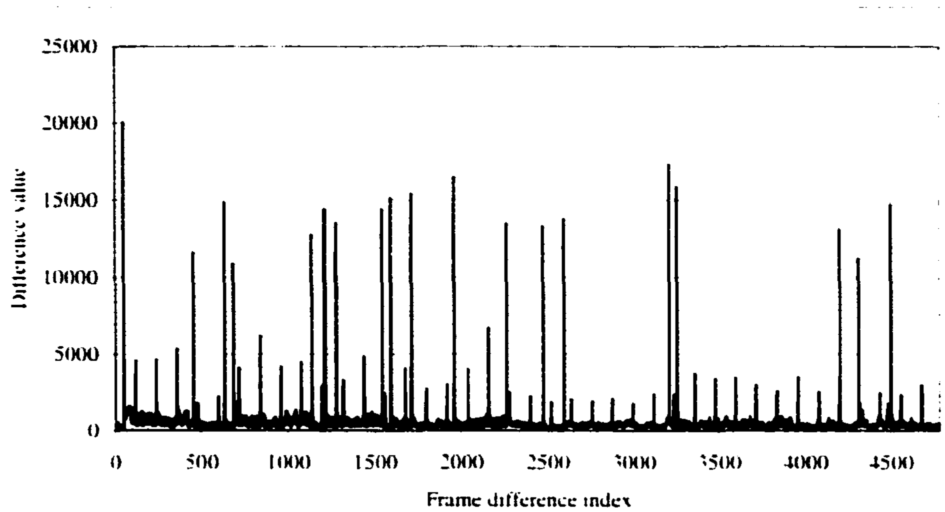

Fig. A.3. The frame difference graph for the conf-discussion video.

A.2 Additional Results Using the Algorithm in Section 3.2.2

Below are the results obtained by applying the algorithm in Section 3.2.2 to two relatively long clips from TABLE 3 called ads and conf-discussion.

TABLE A.3

KFs Selection for the ads Clip Using δ_{al} =125.000 with First-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-65	66	2	0. 22
1	66-97	32	1	66
2	98-258	161	3	98. 164. 230
3	259-304	46	2	259. 296
4	305-407	103	2	305. 357
5	408-437	30	1	408
6	438-579	142	3	438. 506. 559
7	580-614	35	2	580. 608
8	615-650	36	2	615. 640
9	651-674	24	1	651
10	675-758	84	3	675. 714. 755

In TABLE A.4. the number of selected KFs is 71 compared to 26 in TABLE A.2 and this indicates that the problem with small-size clips has been resolved. But another problem appears. that is the bias of the algorithm towards selecting more KFs for longer shots even if they are not very active and this is clear for shots 1. 4. 14. and 16. Some of these shots have moderate activity represented by objects motion and camera works but these activities are not major, see the frame difference diagram in Fig. A.2. On the other hand, reasonable numbers of KFs were selected to represent other shots.

TABLE A.4

KFs Selection for the conf-discussion Clip Using δ_{al} =33.333 with First-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs
0	0-48	49	1	0
1	49-456	408	9	49, 79, 114, 157, 207, 251, 309, 362, 410
2	457-635	179	2	457, 541
3	636-688	53	1	636
4	689-1139	451	9	689, 729, 789, 843, 913, 974, 1031, 1078, 1128
5	1140-1213	74	1	1140
6	1214-1279	66	2	1214, 1275
7	1280-1542	263	5	1280, 1333, 1400, 1450, 1503
8	1543-1593	51	1	1543
9	1594-1713	120	2	1594, 1671
10	1714-1959	246	3	1714, 1799, 1903
11	1960-2262	303	5	1960, 2042, 2131, 2181, 2252
12	2263-2473	211	3	2263, 2323, 2417
13	2474-2595	122	1	2474
14	2596-3208	613	7	2596, 2685, 2780, 2894, 2978, 3080, 3182
15	3209-3255	47	1	3209
16	3256-4202	947	11	3256, 3350, 3432, 3515, 3598, 3685, 3775, 3852, 3929, 4047, 4140
17	4203-4310	108	1	4203
18	4311-4500	190	3	4311, 4373, 4454
19	4501-4782	282	3	4501, 4576, 4677

The bias of the algorithm to select more than the required number of KFs can be noticed in Fig. A.3 that depicts the KFs selected for shot 13 of the tv-show clip. The data in TABLE 13 in CHAPTER III are used to plot Fig. A.4. By investigating that figure, we can observe the same general trend exhibited in Fig. 10 that is the decrease in the number of chosen KFs with the increase in the threshold value.

Fig. A.3. The four key frames selected to represent shot 13 of the tv-show clip. The redundancy is obvious and two KFs would do the work.

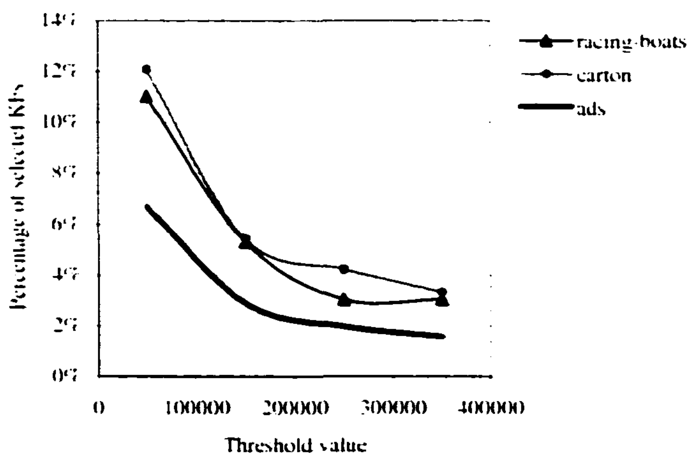

Fig. A.4. Percentage of selected key frames to the total number of frames as function of the threshold value for three different clips using AFS with first-level adaptation.

A.3 Additional Results Using the Algorithm in Section 3.2.3

Below are the results obtained by applying the algorithm in Section 3.2.3 to two relatively long clips from TABLE 3 called ads and conf-discussion.

TABLE A.5

KFs Selection for the ads Clip Using δ_{a1} =125.000 with Second-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs	AI	δ_{a2}
0	0-65	66	2	0. 22	3054	125000
1	66-97	32	1	66	3035	125000
2	98-258	161	2	98. 230	1774	250000
3	259-304	46	2	259. 296	3300	125000
4	305-407	103	1	305	2304	250000
5	408-437	30	1	408	2628	250000
6	438-579	142	2	438. 559	2214	250000
7	580-614	35	2	580. 602	5017	93750
8	615-650	36	2	615. 636	4845	93750
9	651-674	24	1	651	3074	125000
10	675-758	84	3	675. 714. 755	3062	125000

Comparing TABLE A.5 with TABLE A.3. we can notice the effectiveness of the proposed two-level adaptation mechanism in reducing the number of selected KFs in case of long inactive shots. for instance shots 2. 4. and 6. The remaining shots got almost equal numbers of KFs to their corresponding ones in TABLE A.3. Fig. A.5 shows the activity diagram for three shots in the news-cast clip.

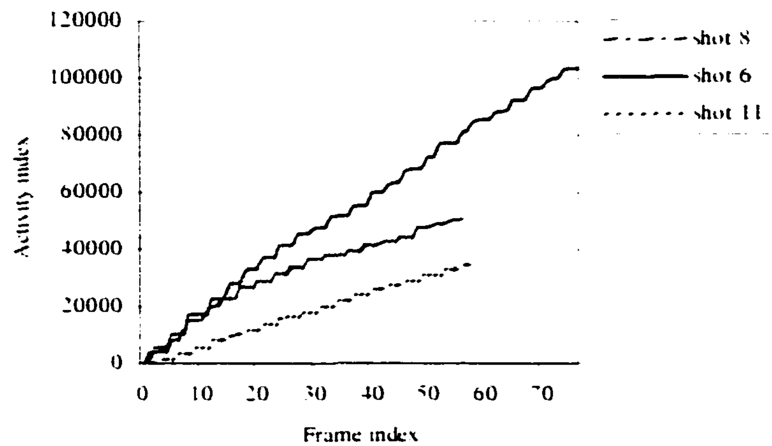

Fig. A.5. The activity index diagram for three shots of the news-cast video.

TABLE A.6

KFs Selection for the conf-discussion Clip Using δ_{a1} =33.333 with Second-level Adaptation

Shot index	Range	# of frames	# of KFs	Index of selected KFs	AI	δ_{a2}
0	0-48	49	1	0	205	66666
1	49-456	408	9	49, 79, 114, 157, 207, 251, 309, 362, 410	722	33333
2	457-635	179	2	457, 633	375	66666
3	636-688	53	1	636	264	66666
4	689-1139	451	5	689, 789, 913, 1030, 1127	605	66666
5	1140-1213	74	1	1140	321	66666
6	1214-1279	66	1	1214	530	66666
7	1280-1542	263	3	1280, 1400, 1503	593	66666
8	1543-1593	51	1	1543	364	66666
9	1594-1713	120	1	1594	449	66666
10	1714-1959	246	2	1714, 1899	353	66666
11	1960-2262	303	3	1960, 2131, 2251	457	66666
12	2263-2473	211	2	2263, 2417	395	66666
13	2474-2595	122	1	2474	217	66666
14	2596-3208	613	4	2596, 2779, 2977, 3180	353	66666
15	3209-3255	47	1	3209	461	66666
16	3256-4202	947	6	3256, 3432, 3598, 3773, 3926, 4137	380	66666
17	4203-4310	108	1	4203	255	66666
18	4311-4500	190	2	4311, 4454	485	66666
19	4501-4782	282	2	4501, 4677	354	66666

In TABLE A.6 many shots such as shots 4, 14, and 16 got reduced numbers of KFs due to their low activities. Other shots got nearly similar numbers to those in TABLE A.4.

A.4 Additional Results Using the ALD Algorithm in Section 3.3

Below are the results obtained by applying the ALD algorithm in Section 3.3 to two relatively long clips from TABLE 3 called ads and conf-discussion.

The sensitivity of this algorithm to lighting conditions is obvious from investigating the results in TABLE A.7 and comparing them with those obtained in TABLE A.5. For shot 0 five KFs were selected as a result of the gradual transition (fad in) occurred at the beginning of this shot. These lighting changes affect the algorithm to choose the first five frames as members of the representative set for that shot. As our algorithm is not handling gradual transitions effects this problem can be tolerated for now. In other shots for instance shots 2, 6, and 10 the number of selected KFs is less than those selected in TABLE A.5.

TABLE A.7

KFs Selection for the ads Clip Using δ_{a1} =12.500 Using ALD Algorithm

Shot index	Range	# of frames	# of KFs	Index of selected KFs	VI	δ_{a2}
0	0-65	66	5	0, 1, 2, 3, 4	1995	12500
1	66-97	32	1	66	437	10000
2	98-258	161	1	98	249	10000
3	259-304	46	2	259, 290	1076	10000
4	305-407	103	1	305	278	10000
5	408-437	30	1	408	626	10000
6	438-579	142	1	438	366	10000
7	580-614	35	2	580, 613	1306	12500
8	615-650	36	2	615, 642	1513	12500
9	651-674	24	1	651	274	10000
10	675-758	84	2	675, 733	622	10000

In TABLE A.8, we can remark one advantage of the ALD algorithm over the one proposed in Section 3.2.3 (AFS) where it chooses less key frames for shots such as shot 1. On the other hand, the AFS algorithm in TABLE A.6 selects 9 KFs to represent the same shot some of them are redundant. On the down side, the effect of the selection criterion is exhibited in other shots in TABLE A.8 in which shots such as shot 4 got more KFs than the necessary numbers compared with those in TABLE A.6

TABLE A.8

KFs Selection for the conf-discussion Clip Using δ_{a1} =3.333 Using ALD Algorithm

Shot index	Range	# of frames	# of KFs	Index of selected KFs	VI	δ_{a2}
0	0-48	49	1	0	267	2666
1	49-456	408	7	49, 77, 199, 240, 304, 358, 424	408	2666
2	457-635	179	1	457	258	2666
3	636-688	53	1	636	365	2666
4	689-1139	451	10	689, 797, 838, 919, 947, 958, 960, 1028, 1080, 1129	407	2666
5	1140-1213	74	1	1140	419	2666
6	1214-1279	66	2	1214, 1246	845	2666
7	1280-1542	263	3	1280, 1440, 1480	254	2666
8	1543-1593	51	1	1543	299	2666
9	1594-1713	120	2	1594, 1680	562	2666
10	1714-1959	246	1	1714	183	2666
11	1960-2262	303	4	1960, 2110, 2158, 2220	389	2666
12	2263-2473	211	1	2263	219	2666
13	2474-2595	122	1	2474	195	2666
14	2596-3208	613	1	2596	113	2666
15	3209-3255	47	1	3209	177	2666
16	3256-4202	947	11	3256, 3333, 3360, 3442, 3480, 3539, 3584, 3600, 3703, 3720, 3790	184	2666
17	4203-4310	108	2	4203, 4282	471	2666
18	4311-4500	190	3	4311, 4408, 4440	365	2666
19	4501-4782	282	3	4501, 4628, 4680	274	2666

VITA

Waleed Ezzat Farag was born in Zagazig. Sharkia. Egypt in 1971. He received his Bachelor of Science with distinct honor in Electronics and Communications Engineering from Zagazig University. Zagazig. Egypt. in May 1993. After that. he worked as an assistant lecturer for the Department of Electronics and Communications Engineering at Zagazig University from 1994 to 1998. During that period. he earned his Master of Science in Computer Engineering in 1997 from the same university. In 1998. Waleed was awarded a full scholarship by Egypt to pursue his Ph.D. in the United States of America. He joined the Ph.D. program at the Computer Science Department. Old Dominion University located in Norfolk. Virginia. United States in August 1998 and passed his comprehensive exam in July 2000. Waleed Farag had published a number of research papers in well known Journals and Conferences in his field. He is a reviewer for a number of IEEE Conferences and a student member of IEEE.